ADVANCED ENGINEERING MATERIALS
www.aem-journal.com

Ontologies for FAIR Data in Additive Manufacturing: A Use Case-Based Evaluation

Thomas Bjarsch, Klaus Drechsler, and Johannes Schilp*

The development of an ontology-based approach for generating Findable, Accessible, Interoperable, Reusable (*FAIR*) data for powder bed fusion, a representative additive manufacturing process, is explored. Addressing key aspects of part design, parameter selection, and processing history, the study identifies both the advantages and disadvantages of using ontologies to manage and utilize distributed and heterogeneous data from additive manufacturing effectively. Critical to this approach is the establishment of unique digital and physical identifiers for physical objects, which facilitate the creation of digital object records and enhance data findability, crucial for enabling digital twins. Despite the benefits of increased findability and domain expandability, challenges persist, such as the complexity of integrating diverse data sources and the high demand for specialized knowledge to navigate ontology-based systems, discussed by incorporating the basic formal ontology. The study also explores data integration techniques using Python, the application of reasoning to reduce manual input, and the implications on reusability. The research demonstrates the potential of *FAIR* data to transform additive manufacturing processes by enabling more efficient data utilization. Applications such as material property and process parameter selection, as well as the creation of digital part records, serve as exemplary implementations showcasing the practical benefits of this approach.

1. Introduction

Additive manufacturing (*AM*) is utilized in industries such as aerospace and medicine, serving design freedom, eco-nomical production for small batches, and function integration, as exemplary shown in **Figure 1** with a multimaterial. Addressing lean production, enhanced material properties, and traceability of manufactured goods necessitates the analysis of a broad dataset encompassing the manufacturing process and its environment. A primary challenge of such analysis is that data from various sensors, machines, and vendors are dispersed across multiple sources, such as files, databases, and cloud storages, each in differing formats.

Each storage type and format requires specific access methods and contextual human knowledge to transform data into usable information. For example, correlating the manufacturing direction with material properties requires accessing the machine job file, which contains the part orientation linked to a machine-specific part ID. This part ID is often different from the IDs used for the manufactured part or testing results. When a domain expert, such as a machinist, leaves the institution, the contextual knowledge needed to interpret the data, either manually or through algorithms, is often lost. Over time, many links between materials and processes may no longer be established, leading to the loss of valuable knowledge. Additionally, the vast number of potentially relevant factors, the generation of several hundreds of gigabytes of data in a powder bed fusion (*PBF*) process, and the involvement of multiple domains complicate data analysis methods aimed at achieving optimization objectives.[1,2] Consequently, data scientists spend about 80% of their time finding, filtering, reformatting, and integrating data.[3,4]

Recently, there has been growing interest in the *FAIR* principles, which demand findable, accessible, interoperable, and reusable data to overcome the gap between information requirements and heterogeneous data sources.[5] The present study investigates how ontologies, which offer human understandability through their semantics, while their formalized syntax enables machine processability, can be applied in *AM* to adhere to *FAIR* principles. The primary goal is to formalize data from heterogeneous sources. This is done by describing it with domain-specific terms and integrating it into existing models,

T. Bjarsch
Department for AI and Digital Engineering
Fraunhofer Institute for Casting, Composite and Processing
Am Technologiezentrum 2, 86159 Augsburg, Germany
E-mail: thomas.bjarsch@igcv.fraunhofer.de

K. Drechsler
Chair of Carbon Composites
Technical University of Munich
Boltzmannstr. 15, 85748 Garching, Germany

J. Schilp
Chair of Digital Manufacturing
Augsburg University
Eichleitnerstr. 30, 86159 Augsburg, Germany

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/adem.202401528.

DOI: 10.1002/adem.202401528

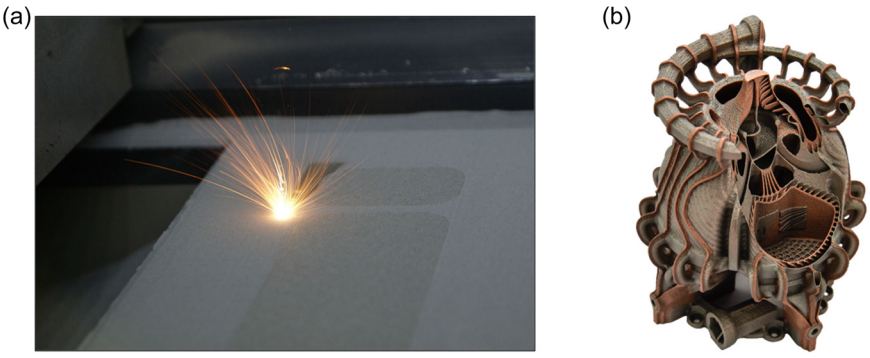

Figure 1. a) The laser in the powder bed fusion process locally melts the powder to successively build the part. b) Additively manufactured aerospike engine made from tool steel and copper alloy. Reproduced with permission.[52]

namely the Basic Formal Ontology (BFO) and Common Core Ontologies (*CCO*). The result is a comprehensive process and material library that enhances part design and enables part traceability. Using data from PBF processing of metals, representative for *AM*, a prototypical ontology-based data management system is created. The implementation details, advantages, disadvantages, application hurdles, and perspectives to use ontologies for *FAIR* data generation are discussed by answering subsequent research questions: 1) F: Which aspects must be considered for data and physical objects like printed parts to be identifiable using ontologies? 2) A: How can the workflow to access heterogeneous data sources be implemented using ontologies? 3) I: What are the advantages and disadvantages of using the *BFO* as a widely used top-level ontology (*TLO*) for enforcing interoperability? 4) R: How can the *FAIR* structured data be reused for applications like process parameter selection, part design, and a digital part record?

2. Theory and Literature Review

2.1. Ontologies: Overview

Combining popular definitions by Gruber (1993) and Borst (1997), Studer et al. defined an ontology as a "[...] formal, explicit specification of a shared conceptualization [...]".[6–8] Here, conceptualization refers to "[.] an abstract, simplified view of the world that we wish to represent for some purpose [...]".[9,10] This indicates that an ontology represents entities from a specific domain and is created to serve a particular purpose. The outcome of the conceptualization is a set of concepts, also known as classes or universals, arranged in a hierarchy, called a taxonomy.[10,11] Each class is defined by "[...] entities in reality that are responsible for the structure, order, and regularity [...]".[11] Thus, a class is defined based on the common characteristics of its specific individuals.

For example, in the context of materials science, the taxonomy could include Entity–Material–Steel–Stainless Steel. A specific plate of stainless steel in a company's production line would be an individual then. In addition to classes, relationships between classes and individuals need to be defined to describe their interactions.[10] The term shared in the definition indicates that the defined concepts must follow a consensus among multiple parties, rather than representing an isolated view.[10] This is especially important when concepts and data need to be shared across multiple domains, as in the present approach to link heterogeneous data sources for sharing both within institutions and with the wider materials science community.

The phrase explicit specification emphasizes that a set of vocabularies is used to represent each class, and axioms are employed to constrain the interpretation of these vocabularies.[10] Formal means that all expressions must be in a machine-readable format, distinguishing them from natural language.[10] A widely adopted modeling language for ontology formulation, used in this study, is the W3C-standardized web ontology language (*OWL*).[12] All expressions are formulated as triples, consisting of a subject, predicate, and object. For instance, a triple stating a hierarchical arrangement is *NonCorrosiveSteel subClassOf Steel*. The complete ontology is then built as the sum of all such triples.

For further clarification the terms data, information, knowledge, and wisdom need to be distinguished. Following widely accepted work of Ackoff, data are "[...] symbols that represent properties of objects, events, and their environment [...]".[13] Information makes data useful through analysis in various aspects, addressing questions like "[...] who, what, where, when and how many [...]".[13] Knowledge is the "[...] transformation of information into instructions [...]" and wisdom "[...] is the ability to increase effectiveness [...]".[13] Ontologies, in turn, provide semantics and relationships between data to form information, create knowledge by classifying this information, and enable wisdom through the use of knowledge and reasoning mechanisms.[14]

2.2. BFO and CCO

The objective of using ontologies in this work is to formalize and structure heterogeneous data sources to make them *FAIR*. *FAIR* principles demand interoperability, which can be addressed using a vocabulary for class definition that is shared across multiple domains, as described by Studer et al. and by reusing existing classes.[8,11] This seeks to avoid the development of isolated ontologies, which shifts the problem of information exchange to a formalized level. One approach is to build upon an existing *TLO*, which serves abstract classes that are further

detailed during the development process with the classes needed for the specific application. If multiple institutions use the same *TLO*, separately developed ontologies can be easily merged. Following the same principles, midlevel ontologies provide more detailed classes based on an existing *TLO*.

The present approach reuses *BFO* as a *TLO* for three reasons: first, *BFO* is *ISO* standardized according to *ISO/IEC 21838-2:202*. Second, it is used by more than 350 research projects, which enhances interoperability.[15] Third, the underlying research project is part of the MaterialDigital initiative, which seeks to harmonize data and information exchange across the entire materials science community.[16] The initiative's midlevel ontology, currently under development, will be based on *BFO*, making future integration easier. A preceding evaluation showed that due to the high reusability of existing classes for the present use case and the available documentation, the *CCO* is considered to be suitable to serve as mid-level ontology (*MLO*).[17,18]

The *BFO* currently contains 36 classes at an abstract level.[15] Exemplary classes include Process, Object, and Quality. Reusing the classes from *BFO*, *CCOs* further detail concepts like *ActOfManufacturing* or *PortionOfMaterial*, which can be reused for the current approach. For a detailed description of the class structures of *BFO* and *CCO* readers are referred to literature.[11,18] The beneficial and disadvantageous aspects of both decisions are discussed in Section 4.2.

2.3. FAIR Principles

The *FAIR* principles, proposed by Wilkinson et al. initially addressed data management in academia to enhance long-term reusability, but have since been adopted and demanded for data management in industry-related fields.[4,5] They require that data be findable, accessible, interoperable, and reusable.

To be findable, data must have a globally unique and persistent identifier, be described with a relevant set of metadata containing the identifier itself, and be registered in a searchable index.[5] Examples of addressing findability include the Digital Object Identifier, commonly used to identify scientific papers, or HTTP URLs, as used in platforms such as FAIRDOM, a platform for systems biology.[19,20] Accessibility can be achieved if the data can be retrieved using a standardized communications protocol that is open, free, and universally implementable. Each protocol must also provide authentication and authorization procedures.[5] Again HTTP is an example, as a widely used protocol that fulfills accessibility requirements and is commonly used for web browsing.[21] To fulfill interoperability, data must use a formal, accessible, shared, and broadly applicable language for knowledge representation. The vocabularies must follow FAIR principles, and metadata must clearly reference the main data object.[5] Languages supporting interoperability include resource description framework (RDF), OWL as used in this work, and JavaScript Object Notation for Linked Data (JSON-LD).[21] As mentioned in the previous section, interoperability can be enhanced by reusing existing classes, that is, vocabularies from TLOs, as done in the current work by reusing BFO and CCO. To be reusable, data must be described with relevant attributes, an accessible usage license must be provided, detailed provenance must be given, and domain-relevant community standards must be met.[5] Examples of meeting these criteria include clearly describing the scope of the data, providing labels, using licenses like MIT or Creative Commons, and adhering to common file formats.[21]

In summary, data should be findable using rich identifiers and metadata, accessible via standardized protocols, interoperable through semantics, and reusable with accurate attributes. Discussions on meeting the FAIR principles with an ontology-based approach for AM are provided in Section 4.

2.4. Ontologies and Data Management in AM

In a literature review, recent relevant works dealing with ontologies and data management in the context of *AM* have been analyzed and categorized by the year, main application, used modeling language, if they developed a workflow or tool, incorporated a substantial amount of material or process data, and whether they reused a *TLO*. The results are given in **Table 1**. Addressed process domains include *AM*, *PBF*, electron beam melting (*EBM*), fused filament fabrication (*FFF*), material extrusion (*MEX*), direct energy deposition (DED), and lithography (*LIT*) and therewith a wide band of applications. Used languages for data and information modeling are predominantly *OWL*, but further include extensible markup language (*XML*), relational databases (*DB*), semantic application design language (*SADL*), and business process model and notation (*BPMN*).

For *AM*, several data models have been developed in recent years. Mohd et al. are the only ones to develop an *AM* ontology reusing *BFO* and *CCO*.[22] They modeled several processing steps and process factors but applied them to only a small set of experimental data for demonstration purposes. The ontology itself has not been found to be publicly available. Therefore, in the current work, only a small set of generic terms is reused and further detailed for the PBF process and material characterizations.[22] Another data model was developed by Liu et al. comprising a comprehensive set of product qualities, postprocessing parameters, process signatures, process parameters, and design parameters to capture experimental data, but lacking in reusing a *TLO*.[23] Several of their concepts have been adopted for the current work and mapped to *BFO*. In collaboration with more than 100 *AM* experts, Kuan et al. developed a common data model, from which 395 terms have been *ASTM* standardized in an *AM* common data dictionary. Additionally they evaluated the mapping of exemplary processing data using XML- and graph-based approaches.[24] Li et al. proposed a data model for describing information related to material, process, simulation, and measurement data, which are linked by IDs, and created 55 classes.[25] Except for the publication by Mohd et al. none of the data models reused an existing *TLO*, which is essential for enhancing interoperability according to *FAIR* principles. Furthermore, a detailed discussion of the benefits and disadvantages of using *BFO* and *CCO* in the context of *AM* data management is considered to be missing.

Categorizing by main applications shows multiple perspectives. One application of the created data models includes defect detection. For detecting new qualitative correlations between processing factors and occurring defect types, Wang et al.

Table 1. Overview of recent publications using ontologies in the context of AM.

References	Year	Domain	Main application	Language	Workflow	Mat.	Proc.	TLO
Li et al.[25]	2024	PBF	Data models for AM projects	XML	–	✓	✓	–
Kuan et al.[24]	2024	AM	Development of a shared common data dictionary & model	XML/OWL	–	–	–	–
Wang et al.[26]	2023	AM	Detecting relationships between process factors and defect types	OWL	–	–	–	–
Huang et al.[36]	2023	AM	Evaluating AM machines by cost, strength, etc.	OWL	–	–	–	–
Grandvallet et al.[53]	2023	EBM	Formulation of processing steps and action rules	–	–	–	–	–
Bonello et al.[54]	2023	PBF	Defect forecasting	–	–	–	✓	–
Formentini et al.[29]	2022	AM	Evaluation of the suitability of AM technologies using design guidelines	OWL	–	–	–	–
Huang et al.[37]	2022	FFF	Optimization of part orientation	OWL	–	–	–	–
Liu et al.[23]	2022	PBF	Data management, defect detection, digital twins, correlation analysis	DB	✓	✓	✓	–
Jarrar et al.[34]	2022	PBF	Cost calculation for part manufacturing	OWL	✓	✓	✓	–
Kulchin et al.[38]	2022	DED	Database for equipment and materials	OWL	✓	–	–	–
Haruna et al.[30]	2022	FFF	Support for part design	OWL	✓	–	–	–
Gmeiner et al.[55]	2022	LIT	Retrieving qualitative material properties	OWL	–	–	–	–
Ahn et al.[56]	2022	MEX	Influencing factors on surface roughness and support reduction	OWL	–	✓	✓	–
Roh et al.[27]	2022	AM	Selection of sensors for defect detection	OWL	–	–	–	–
Mayerhofer et al.[31]	2021	AM	Evaluation of part manufacturability	OWL	✓	–	–	–
Guo et al.[39]	2021	FFF	Proposal for data linking via cloud application	DB	✓	–	–	–
Rojek et al.[40]	2021	AM	Correlation of parameters and material properties	DB	✓	✓	✓	–
Ko et al.[32]	2021	PBF	Derivation of design rules with ML	OWL	✓	–	–	–
Jarrar et al.[35]	2021	PBF	Cost calculation for manufacturing	OWL	–	–	–	*DOLCE*
Chen et al.[57]	2021	PBF	Detection of relationships between parameters and mech. properties	OWL	–	✓	✓	–
Roh et al.[58]	2021	AM	Combination of process parameters and physics models	OWL	–	–	–	–
Roh et al.[28]	2021	PBF	Selection of sensors for defect detection	OWL	–	–	–	–
Belkadi et al.[59]	2020	PBF	Process model for overview	BPMN	–	–	–	–
Kumar et al.[41]	2019	AM	Data correlation with Machine Learning	SADL	✓	✓	✓	–
Aggour et al.[2]	2019	PBF	Ontology-based data management	SADL	✓	✓	✓	–
Leuschitz et al.[60]	2019	FFF	Architecture for cloud-based manufacturing	OWL	✓	–	✓	–
Qian et al.[61]	2019	FFF	Architecture for cloud-based manufacturing	XML	–	–	✓	–
Han et al.[33]	2019	FFF	Support for machine selection	OWL	–	–	–	–
Xiong et al.[62]	2019	DED	Evaluation of part manufacturability	–	✓	–	–	–
Lepuschitz et al.[63]	2019	AM	Evaluation of part manufacturability	OWL	–	–	–	–
Mohd et al.[22]	2019	AM	Process model for overview	OWL	–	–	–	*BFO*
Moges et al.[64]	2019	PBF	Uncertainties in model predictions	OWL	–	–	–	–

used an ontology in conjunction with a relational enhanced graph convolutional network, for example, for detecting a potential correlation between the staircase effect and porosity.[26] Roh et al. provided an ontology-based system supporting the selection of appropriate sensors for defect detections.[27,28] By formulating design guidelines and applying them to several part geometries, Formentini et al. optimized their part design for several AM technologies using an ontology-based approach.[29] Haruna et al. modeled a comprehensive set of design guidelines for FFF, such as minimum wall thickness, and implemented a prototypical software tool to evaluate the manufacturability of a given designs.[30] Mayerhofer et al. developed a tool where an ontology captures the capabilities of processes and printers to evaluate the manufacturability of a given part.[31] Ko et al. created design rules using artificial intelligence to evaluate the manufacturability of parts.[32] By modeling the capabilities of 3D printers and 3D-part properties, Han et al. presented an approach for selecting suitable machines.[33] Modeling manufacturing steps

and cost factors, Jarrar et al. created a workflow and tool to estimate the costs for part production in PBF, reusing the TLO DOLCE.[34,35] Huang et al. proposed a multiattribute three-way decision-based approach in conjunction with an ontology containing information about machine types, such as build times and costs, to select an appropriate machine for a given task.[36] Huang et al. also successfully optimized part orientations to improve characteristics like volumetric errors for FFF using ontology-based reasoning approaches.[37] None of the publications address the use of an ontology-based data management in AM to provide a digital part record, which tracks the history of the manufactured part along the process chain. Outlining the necessary requirements to achieve this, an application scenario highlights the benefits of FAIR data management in the present work.

Several approaches to data management in *AM* have been investigated. Liu et al. introduced a cloud digital twin system designed to collect product lifecycle data from distributed edge devices, that focus on a product or process at a specific lifecycle stage on the shop floor.[23] The two systems communicate effectively, enhancing data availability throughout the product lifecycle. The developed software uses commercial software MaterialCenter from MSC Software, providing data visualization along with file upload and download capabilities, but which might be seen as unFAIR due to missing open-source solutions. The platform utilizes DB to store, structure, and access distributed sensor data, such as the current machine status and monitoring data. It also incorporates a variety of material properties, which enables defect detection as a result. Due to the nature of databases, a comprehensive data model describing the relations between the individual factors is missing, but some factors of Liu et al. have been incorporated in currents work. Kulchin et al. designed a framework and knowledge database that consolidates information about equipment used for *DED*, incorporating consumable materials and technological operations.[38] Guo et al. outlined a multilayered cloud-based system for *FFF*, enabling monitoring of an *FFF* machine through a prototype cloud application.[39] Rojek et al. developed software for training artificial neural networks using a database containing process and material data to assist in material selection and predict, for example, the tensile strength.[40] However, a FAIR data model was not provided, as they used single databases. Kumar et al. developed a software abstraction called *NodeGroup* at GE to map existing databases containing sensor and processing information to nodes, which were then used with machine learning algorithms to establish correlations between time series data and resulting part properties.[41] Described more in detail by Aggour et al. a knowledge graph, written in *SADL*, connects CT scans with manufacturing parameters and monitoring data. This approach facilitated the correlation of fatigue properties and process parameters through feature and label selection, with machine learning algorithms running in the background.[2] Current work partially adopts parts of the main architecture, as using a user interface for data ingestion and querying and a knowledge layer, linking the heterogenous data sources, but incorporating reasoning capabilities and usage of *BFO* and *CCO* to address aspects of *FAIR* principles.

2.5. Research Gap and Paper Structure

Although publications have beneficially utilized ontologies in narrowly defined application scenarios, there is a noted lack of comprehensive incorporation and examination of applying *FAIR* principles in practice, as recently emphasized by the *NIST* Institute.[1,4] A critical review of the entire process, from implementation to the usage of ontologies for the practical application of *FAIR* principles in *AM*, is missing. Therefore, the paper structure is oriented according to the full process from development to usage to address this gap. Before discussing the results, Section 3 initially introduces the methods of data collection and the materials used.

Findability, as outlined by the FAIR principles, needs to be addressed for the data itself, but for the use case of a digital part record provided in this work, findability requires identification at both the digital and physical levels. The application of a digital part record using ontologies and the theoretical background has not been explored in depth for this scenario. This is elaborated upon in Section 4.1.

Interoperability at a high level and across domains requires the reuse of classes and vocabularies from existing top- and midlevel ontologies, as mentioned in Section 2.2. As shown in Section 2.4, available data models, except for the work of Mohd et al. have been developed without incorporating a suitable *TLO*, thereby hindering interoperability in achieving *FAIR* principles. Therefore, the current work's data model is built upon *BFO* and *CCO*, with their practical advantages and disadvantages presented in Section 4.2.

Drawing from the heterogeneity of data demonstrated by Liu et al. and Aggour et al. elements from the data management structure of Aggour et al. and the data model by Liu et al. have been reused in the implemented ontology-based data management system.[2,16,23] Theoretical and practical perspectives on the requirements, strengths, and limitations for data linkage within such a system are highlighted in Section 4.3.

Methods to enhance efficiency in *FAIR* data generation using reasoning and Python-based inference for data management in *AM* are lacking in current publications and are therefore presented in Section 4.4. Finally, the exemplary reuse of *FAIR* data through a material process dashboard and the provision of a digital part record for *AM*, which has not been addressed in current publications using ontologies, are covered in Section 4.5.

3. Methodology

3.1. Data Acquisition and System Boundary Definitions

Applying *FAIR* principles to data structuring prompts the initial question: Which data should be structured? **Figure 2** outlines the main steps to answer this, beginning with the selection and prioritization of relevant data. Since structuring data according to *FAIR* principles incurs initial costs and ongoing maintenance effort, relevance is contextual, leading to the critical question: What is the question or problem for which *FAIR* data provide the answer? From this starting point, two approaches emerge. First, if specific problems to be solved

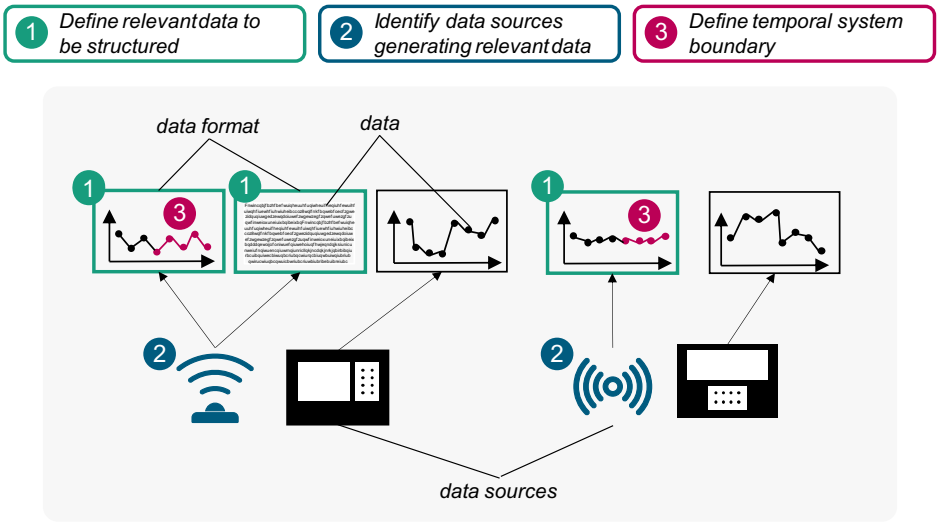

Figure 2. Steps for choosing data to be FAIR: Evaluate relevance, choose data sources, and define temporal system boundaries.

can be clearly formulated, so-called competency questions are used in ontology development.[42] These questions serve as a basis for validation. If the ontology can answer the questions, its purpose is fulfilled. The second approach arises when the questions are not clear a priori. In this case, *FAIR* data can be considered a foundation for addressing future unknown questions or for employing data analysis methods like unsupervised machine learning to discover answers. In this case potentially all data in a desired domain could be relevant. After problem statements are formulated, the next step involves defining system boundaries at a spatial level, such as considering all machines in a particular room or the entire company. Additionally, temporal system boundaries determine the scope of data to be considered, which could include data from the past ten years or only data currently being generated.

Trying to incorporate a reasonable amount of heterogeneous data sources to analyze the applicability of ontologies to generate FAIR data in the present work, three competency questions are formulated addressing typical use cases from engineering and production in industrial practice, as shown in Table 1. They represent a substantial part of the entire AM workflow, starting from part design, continuing through processing with the associated selection of machine parameters, and concluding with the logistics for tracking the manufactured part.

By interviewing domain experts and researching literature, relevant data and their sources for answering the competency questions are summarized in Table 3.[23] The relevance of the captured data is derived from the competency questions in this case. For structural design in first competency question, material properties are essential. A broad set of mechanical properties required for the static and dynamic design of components was chosen based on expert knowledge, incorporating tensile, compressive, and fatigue characteristics. Additionally, thermal properties necessary for part design in environments with significant temperature fluctuations, such as thermal expansion behavior and specific heat capacity, have been included. Based on the second competency question, machine and processing factors that must be specified during manufacturing planning, such as laser power used during processing, are selected. Additionally, factors potentially affecting individual material properties, as identified by expert knowledge, are captured, such as the part's orientation during manufacturing.

The spatial domain for manufacturing is the AM laboratory of the research institute. Limiting the scope to the research institute allows for proper evaluation of data management practices before extending to distributed facilities. Material characterizations are carried out at the research institute and by external project partners. Currently, only data generated during the research project are considered. This approach allows for greater control over data quality and ensures consistency in data collection methodologies, with the provenance of data being clearly identifiable. For historically available data, the necessary context is often missing, which is an argument for using *FAIR* principles, as addressed in Section 4.1.

3.2. Processing and Materials

PBF is an "[...] additive manufacturing process in which thermal energy selectively fuses regions of a powder bed [...]".[43] For *PBF*-LB/M, the energy source is a laser beam *(-LB)* in a metallic material *(/M)*.[43] The recoater creates a thin layer of powder taken from the powder supply. An optical system controls a laser, locally melting the powder to create a 2D contour. After each layer is fused, the build platform lowers, and the process repeats until completion. Once finished, the built part is cleaned of surrounding powder. Material types and product requirements determine post-processing, such as surface finishing or heat treatments. Used *PBF*-LB/M machine is an *EOS M280* from *Electro Optical Systems GmbH*, where 290 specimens are manufactured from steel powder *m4p 316L0.3* and 150 specimens from aluminum powder *m4p PureAl0.1*.

4. Results and Discussion

The architecture for the ontology-based data management system, designed to generate and use *FAIR* data by combining, implementing, and expanding existing works from Aggour et al. and Liu et al. is illustrated in **Figure 3**.[2,23]

The object layer, which includes physical objects such as machines, raw materials, and manufactured parts, is covered in detail with a focus on the findability of data and part identification in Section 4.1. Data generated during *PBF-LB/M* processing is stored in the data source layer. The linkage layer, described in Section 4.3, maps this data to the ontology, which is explained in Section 4.2. The rules engine, discussed in Section 4.4, uses expert-formulated rules to add links and instances, thereby reducing implementation effort. Queries to the ontology and underlying data sources are formulated and executed by the query layer. The front end in the application layer is demonstrated in Section 4.5 and visualizes the part's manufacturing history and process-material correlations, addressing the competency questions presented in **Table 2**.

Table 2. Competency questions and application scenarios for this work.

Competency Question	Application Scenario
What are the material properties (tensile strength, density,...) of parts manufactured with PBF-LB/M?	Structural design
How does a specific machine parameter (e.g., laser speed) affect specific material properties?	Machine parameter selection
What's the manufacturing history of a specific manufactured part, for example, powder charge or machine parameters?	Traceability and logistics

4.1. Object Layer: Findability

Findability demands that data "[...] are assigned a globally unique and persistent identifier [...][,] are described with rich metadata [...] [and] are registered or indexed in a searchable resource."[5] As introduced, present work uses OWL for ontology development. Herein, each instance has its unique International Resource Identifier *(IRI)*, which is enriched with metadata by properties linking it to other instances and being part of a triplestore makes it part of a searchable resource. Using *FAIR* data in productive environments like *AM* requires not only that the data is findable but also that the physical entities the data represent are identifiable. This work distinguishes three mechanisms of identification by incorporating the physical domain, as shown in **Figure 4**.

For *unique digital and physical identification (UDPI)*, the physical entity, such as a printed part, can be uniquely identified using queryable information from the ontology and inspecting them on a physical level (e.g., observing an *ID* on the object or measuring properties). Inspecting characteristics of the physical object and querying them from ontology result in exactly one instance. For *nonunique digital identification (NUDI)*, querying inspected characteristics, such as shapes or an *ID* written on the object, results in multiple instances found in the ontology. This implies that either more distinguishing properties must be used in the query, or the ontology needs to be checked for containing duplicated instances. *Nonunique physical identification (NUPI)* implies that the number of queryable characteristics

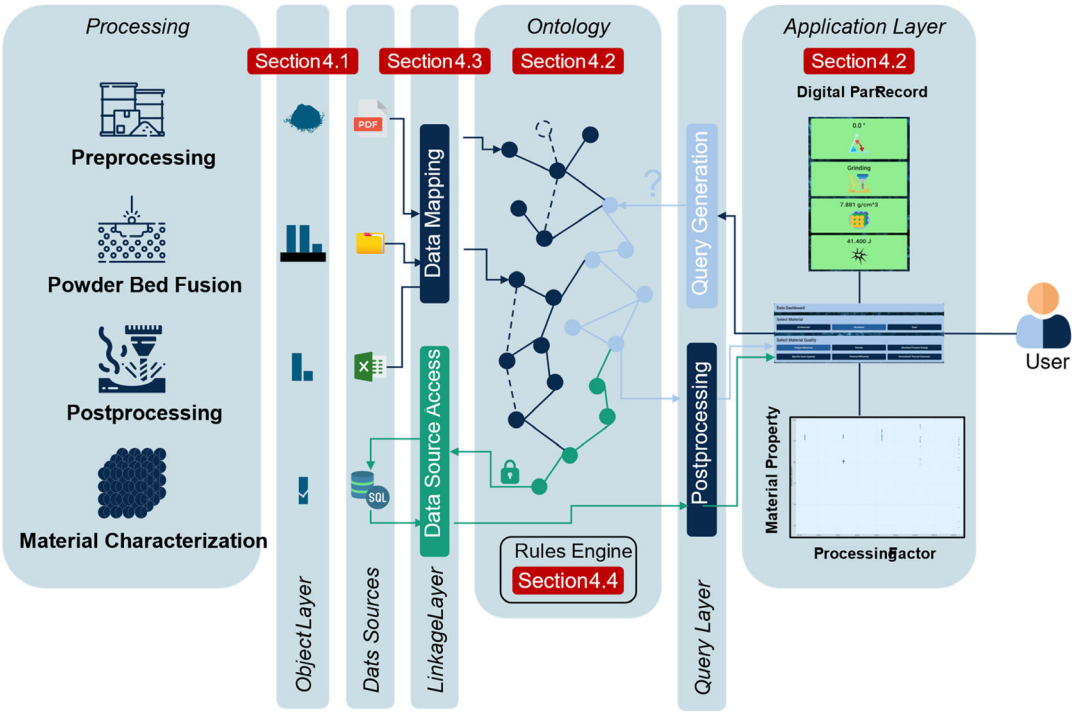

Figure 3. Implemented and analyzed system architecture to enable *FAIR* data generation and usage.[2,23] The digital object record is implemented for printed parts but expandable to all physical objects along the process chain.

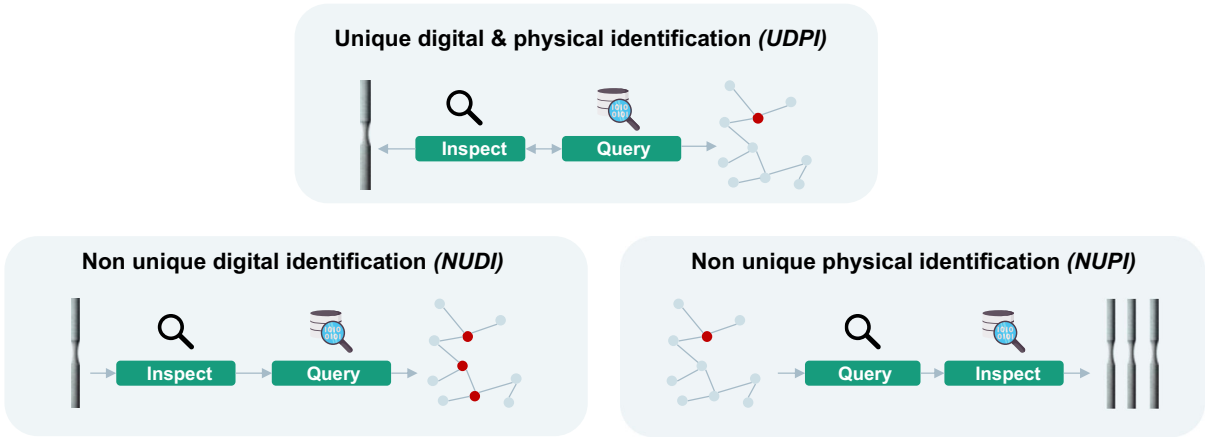

Figure 4. Three mechanisms of identification on a digital and physical level.

is not sufficient to uniquely identify a physical object, as the characteristics match multiple objects.

Two criteria might induce the necessity for *UDPI* of a physical entity. 1) If the application querying and writing *FAIR* data requires the data exchange between the physical object and its instance representing it, as defined for a *digital twin* according to Kritzinger et al. this requires *UDPI*.[44] This is particularly relevant for the digital part record addressed by the third competency question; 2) If the question to be solved and the use case is unknown, a decision must be made based on the expected effort to ensure *UDPI* and the anticipated future necessity for *UDPI*. Coming from *NUDI* or *NUPI* might not enable *UDPI* after physical processing. For example, printed parts with identical shapes and materials may not be distinguishable after they are removed from the build platform.

In other cases, *UDPI* can be omitted in favor of *NUDI* or *NUPI*. For instance, for most applications, it might suffice to know the properties of a specific material class instead of the exact processing history of the specimen made from this material class. When the decision is made to ensure *UDPI* for a physical object and its corresponding instance in the ontology, the necessary steps are outlined in **Figure 5**.

Initially, *sets of characteristics (SC)* that allow for unique identification of a physical object and their *systems of validity (SV)* stating the validity space of each SC_i must be defined. For example, a printed part may be uniquely identifiable by its shape (SC_1), but only when considered alongside all other parts of a build job and if the shape is unique within this build job (SV_1). Another example is a *universally unique ID (UUID)* (SC_2), which is globally unique (SV_2). Subsequently, the SC_i

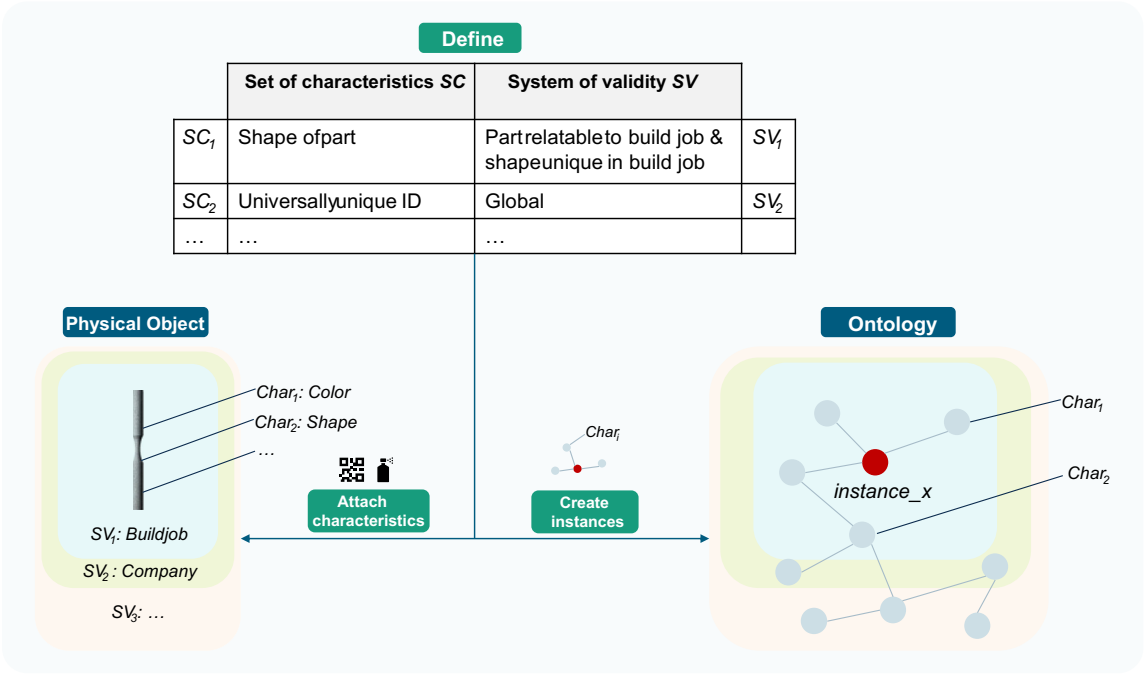

Figure 5. Main steps to ensure UDPI, currently implemented for printed parts.

must be applied to the physical object (e.g., using a marker, sticker, or engravings for the *UUID*) and replicated in the ontology, either by creating new instances or appending to existing instances.

In productive environments SC_i must be processed efficiently. For example, describing an object by stating "the shape is *X*, color is *Y* and it has a scratch of 2 mm at position *Z*" is complex to inspect on both physical and digital level. Reducing SC_i to the minimum results in exactly one characteristic for the object within its SV_i. As it's challenging to find a characteristic that is globally unique for a physical object, the characteristic is simplified as being a unique string identifier.[24]

Three steps are necessary if SC_i is a *UUID* and SV_i is global: The *UUID* must be created, physically attached to the object, and added to the instance representing the physical object in the ontology. The detailed steps involved, both generally and specifically in this work for printed part identification, are illustrated in **Figure 6**.

4.1.1. General View

After creating the *UUID*, the SC_i and SV_i are queried to check if an instance in the ontology already exists, matching the criteria. If so, the *UUID* is linked to the instance. The SC_i and SV_i are inspected for the physical object, and the *UUID* is physically applied. If no instance is found, a new instance with the *UUID* is created, and the physical *UUID* can be arbitrarily applied to an object of the instances class (e.g., to any printed part if the instance is a "part"), as no SC_i on the digital or physical level restricts the assignment. Optionally, further characteristics along with the SC_i and SV_i can be created digitally, potentially necessitating inspection before physical attachment.

4.1.2. Implementation for Printed Part Identification

The right side demonstrates the current implementation aimed at achieving *UDPI* for printed parts. Uploading the build job file generates a new instance for each part in the ontology, each assigned a *UUID* generated according to *RFC 4122* (e.g., *6948DF80-14BD-4E04-8842-7668D9C001F5*), along with the *build job ID*, parameter set name, part coordinates, and an *integer ID* (SC_i) unique within each *build job* (SV_i). A printable *QR*-Code encoding the *UUID* and the *integer ID* is generated in the backend. After *PBF-LB/M* processing, each part is manually marked with the *integer ID*, uniquely identifying it within each build job. Following erosion from the build platform, the integer variable written on the part, along with the printed *QR* code, is used to attach the 128-bit *UUID* to the part. The utilized 128-bit *UUID* minimizes the likelihood of duplicate identifiers almost to 0, making it robust across global contexts. Practically, only the first eight digits are assumed to be unique within the current scope and queries, simplifying identification tasks like noting part numbers on a test sheet. This truncated form is sufficient for smaller contexts, such as a single company, whereas the entire *UUID* may be necessary for global data sharing.

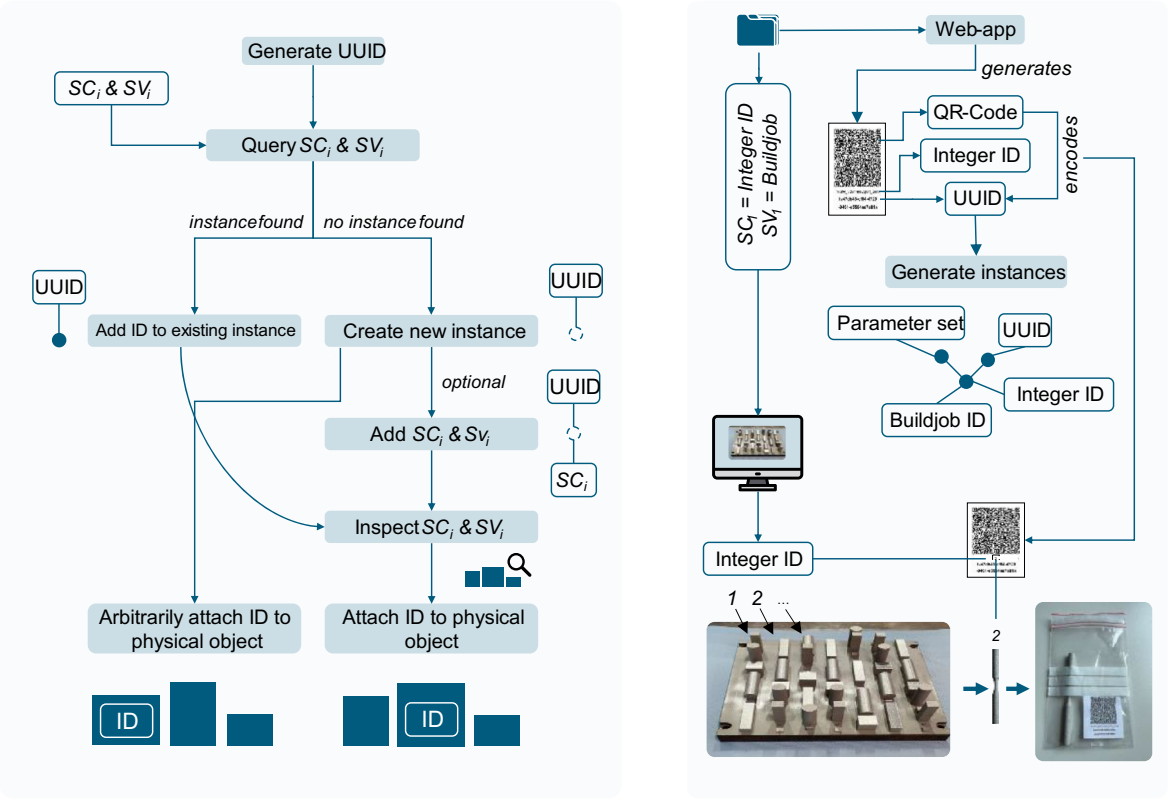

Figure 6. General steps necessary to ensure *UDPI* with a *UUID*. Right: Specific implementation using a web app to ensure *UDPI* for printed parts.

4.2. Ontology Design: Interoperability

Interoperability necessitates that "[...] data use a formal, accessible, shared, and broadly applicable language for *knowledge* representation [...][,] use vocabularies that follow *FAIR* principles [...] [and] include qualified references to other (meta)data."[5] Addressing the first issue, widely used and *W3C*-compliant *OWL* is proposed.[12] *OWL's* formal syntax uses triples of *subject*, *predicate*, and *object* to describe *classes*, *instances*, their *relationships (object properties)*, and data values (data *properties)*. The working principles are detailed in the specifications.[12]

For vocabularies, *FAIR*-compliant *OWL* ontologies are achievable using a publicly available *IRI* for each instance and class (e.g., as used for this work: *https://w3id.org/ODE_AM*), which ensures findability (via human-readable definitions and labels), accessibility (public access), and reusability. The requirement to "[...] include qualified references to other (meta)data [...]" translates in *OWL* ontologies to adopt existing *TLOs* and *MLOs*, which offer abstract-level classes and comprehensive documentation. The selection of a *TLO* is influenced by use-case requirements, personal experience, and its applications across other fields. As introduced in Section 2.2 the *ISO*-standardized *BFO* as*TLO* and CCO as *MLO* are considered.[17,45]

In terms of *FAIR* data, the ontology classifies each data point *(e.g., :instance_x rdf:type :Density)* and uses *object properties* to represent relationships between entities in the physical world *(e.g., :density_x obo:quality_of :part_x)*. As they are set in context now and questions such as "[...] who, what, where, when and how many [...]" are enabled, data are transformed to information, following the definition of *Ackoff*.[13]

Reusing a *TLO* and *MLO* requires the instantiation of an existing class in the *TLO* or *MLO*, the creation of subclasses, and the definition of which *object* and data *properties* to use between instances and data values. The steps for modeling planning applied and discussed in this work are in **Figure 7**.

For *relevant* data previously summarized in **Table 3**, a glossary is developed based on workshops and expert interviews to define terms describing the entities. The definitions of modeling patterns, that is, how terms from the glossary can be represented with triples of instances, *object* and *data properties* and data values, are documented graphically, serving as a blueprint to implement importers and query generators.[46] *BFO* and *CCO* documentation, discussion groups, existing labels, and examples provided in *BFO* and *CCO* guide the definition of modeling patterns.[47]

The limitations of this approach include several key issues. One challenge is the isolation of the glossary, where terms in the glossary are duplicated as entities in the ontology. Sustainable expandability requires a collaborative approach that supports versioning and validation. Another challenge is the manual transfer of graphical diagrams of modeling patterns to importers, which results in duplicated entities and complicates maintenance while being time-consuming. More sophisticated, automatic transfer mechanisms, such as *Chowlk*, are needed for industrial-scale applications. Additionally, complex diagrams without effective filter mechanisms make it difficult to handle and understand the content, as they may contain hundreds of instances and require in-depth knowledge of the underlying modeling structure. Furthermore, the learning curve for understanding modeling patterns proposed by *BFO* and *CCO* is steep, demanding significant effort and expertise, which poses difficulties for domain experts unfamiliar with knowledge modeling.

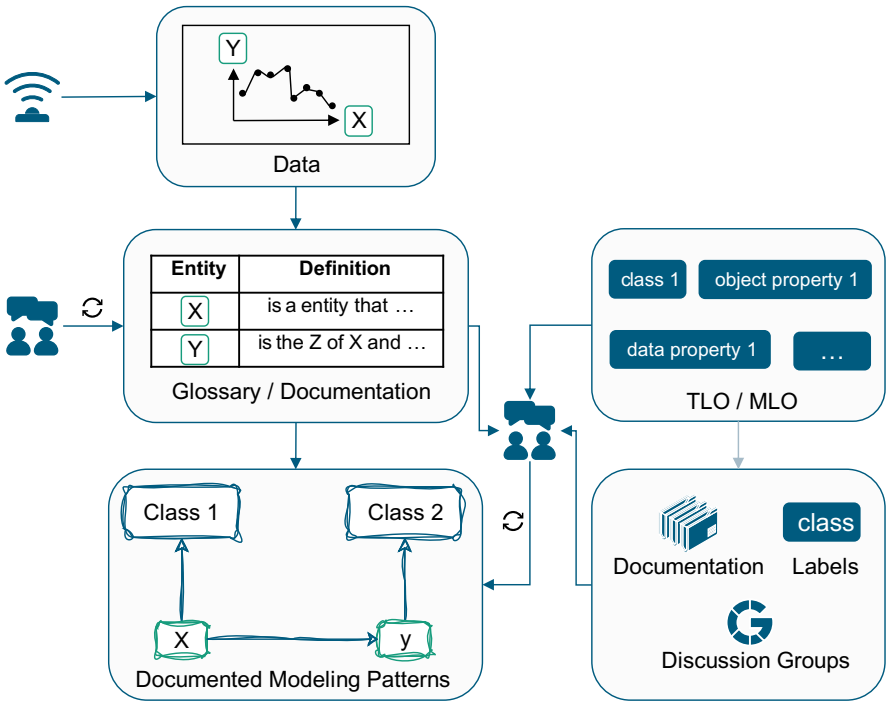

Figure 7. Steps involved to create modeling patterns reusing existing classes from MLO and TLO.

Table 3. Excerpt of relevant data to answer competency questions for this work.

Category	Entities	Data Source	Format
Machine parameters	e.g., laser speed	Build job-file	openjz
Part qualities	Build orientation	Human knowledge	–
	Geometry	STL-file	stl
	Build platform coordinates	Build job-file	openjz
Powder characteristics	Material type, chemical composition, Hall flowrate	Product data sheet	pdf
	Particle size distribution	Testing machine	xlsx
Postprocessing method	Turning, milling, etc.	Human knowledge, UWB system	sql
Material properties	Compressive strength	Defined spreadsheet	xlsx
	Tensile strength	Defined spreadsheet	xlsx
	Specific heat capacity	Testing machine	xlsx
	Thermal expansion	Testing machine	xlsx
	Impact energy	Testing machine	xlsx
	Density	Testing machine	xlsx
	Fatigue properties	Testing machine	xlsx

There is also variability in interpreting modeling guidelines, as there is no universally "correct" way to ensure *BFO* and *CCO* compliance, shifting the problem of nonshareable semantics from natural language to a formalized level. Finally, rigid modeling patterns, while reducing the effort needed to formulate queries and enhancing shareability, create overhead by potentially requiring the creation of instances that do not carry valuable information.

While it is beyond the scope of this article to discuss each modeling pattern used to represent data from Table 3, **Figure 8** highlights the patterns that are extensively and beneficially used, with *object* and data *properties* omitted for clarity.

Identified advantages of modeling patterns following *CCO* and *BFO* for entities from Table 3 include the following.[18] 1) *Specified behavior: cco:PerformanceSpecification* allows to differentiate between prescribed and actual occurrences. For example the parameter "laser speed" defines the *intended* speed of the laser, its *actual* speed might vary; 2) *Object identification: cco:CodeIdentifier* in conjunction with *OWL functional cco:designates* and *nonfunctional cco:is_about* allows automatic identification per defined SC_i. For example, linking two instances to the same *UUID* with *cco:designates* infers with the reasoner that both instances are the same *(owl:same_as)*. Modeling a nonunique ID using *cco:is_about* (e.g., "2") doesn't allow this inference; 3) *Temporal assignment: obo:*

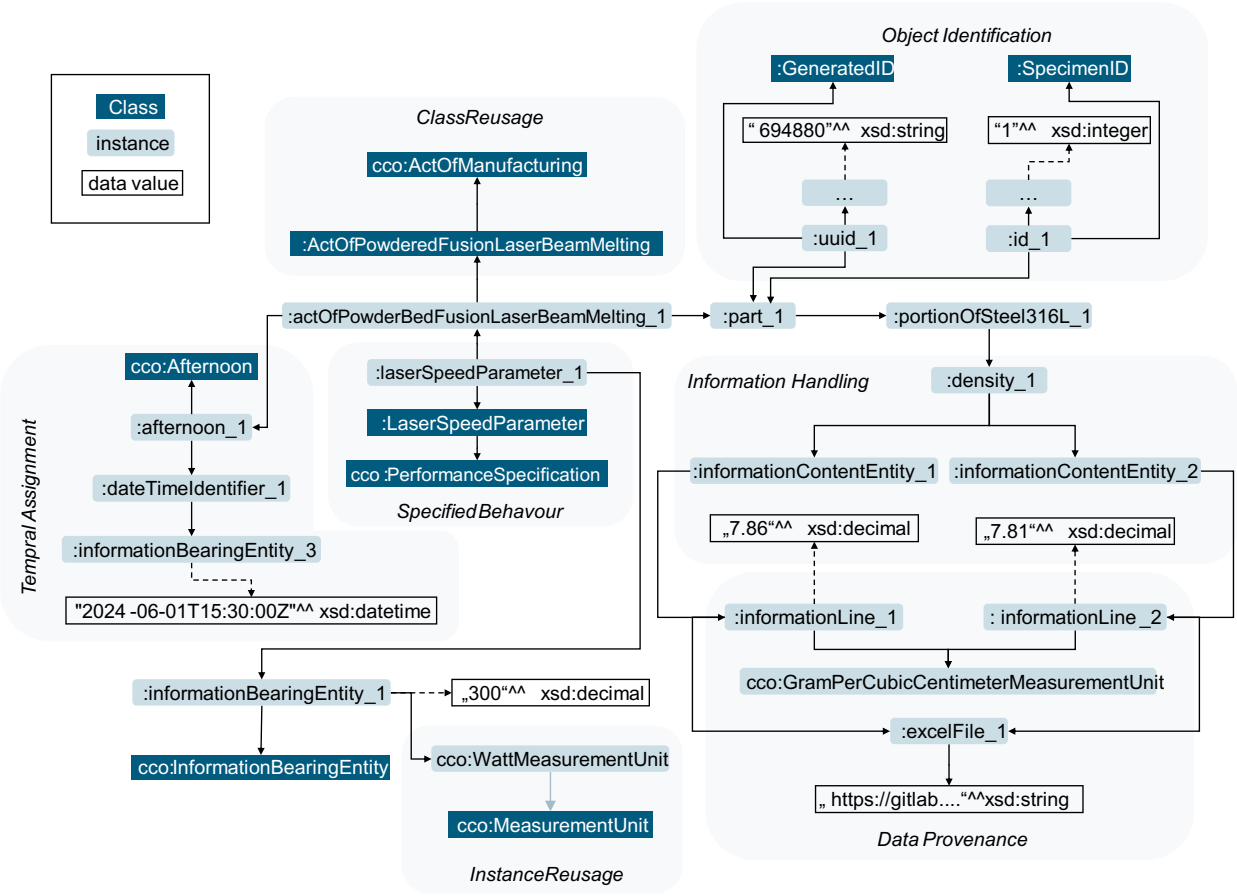

Figure 8. Beneficial aspects for extensively used modeling patterns from *BFO* and *CCO* for domain *AM*.

OneDimensionalTemporalRegion in conjunction with *cco:has_starting_instant* and *cco:has_ending_instant* allows the description of timespans, start and end times for processes, and facilitates sequence modeling; 4) *Entity* and *information* distinction: Differentiating between an entity (e.g., *:density_1*) and information about this entity allows multiple measurements of the same quality from various sources, potentially yielding different results. For example, a density might be measured twice with different results; 5) Information content and bearer distinction: Distinguishing between information (*cco:InformationContentEntity*) and the bearer of this information (*cco:InformationBearingEntity*) allows to comprehend the source of an information (e.g., a excel file stored in the cloud), supporting *findability* and *accessibility*.

4.3. Linkage and Data Source Layer

The linkage layer mapped the data from their sources and generated un-*FAIR* data using created modeling patterns from the blueprint discussed in Section 4.2, following the steps and architecture outlined in **Figure 9**. An alpha version web application is hosted on the institute's computing cluster. It features an *Angular* frontend, *Python* backend, *GraphDB* triplestore, utilizing *Gitlab* for versioning and as a cloud repository for demonstration purposes.

Creating *FAIR* data using ontologies requires the import of data, so the import mechanisms must be automatized as far as possible to reduce necessary user interaction and lower application hurdles.

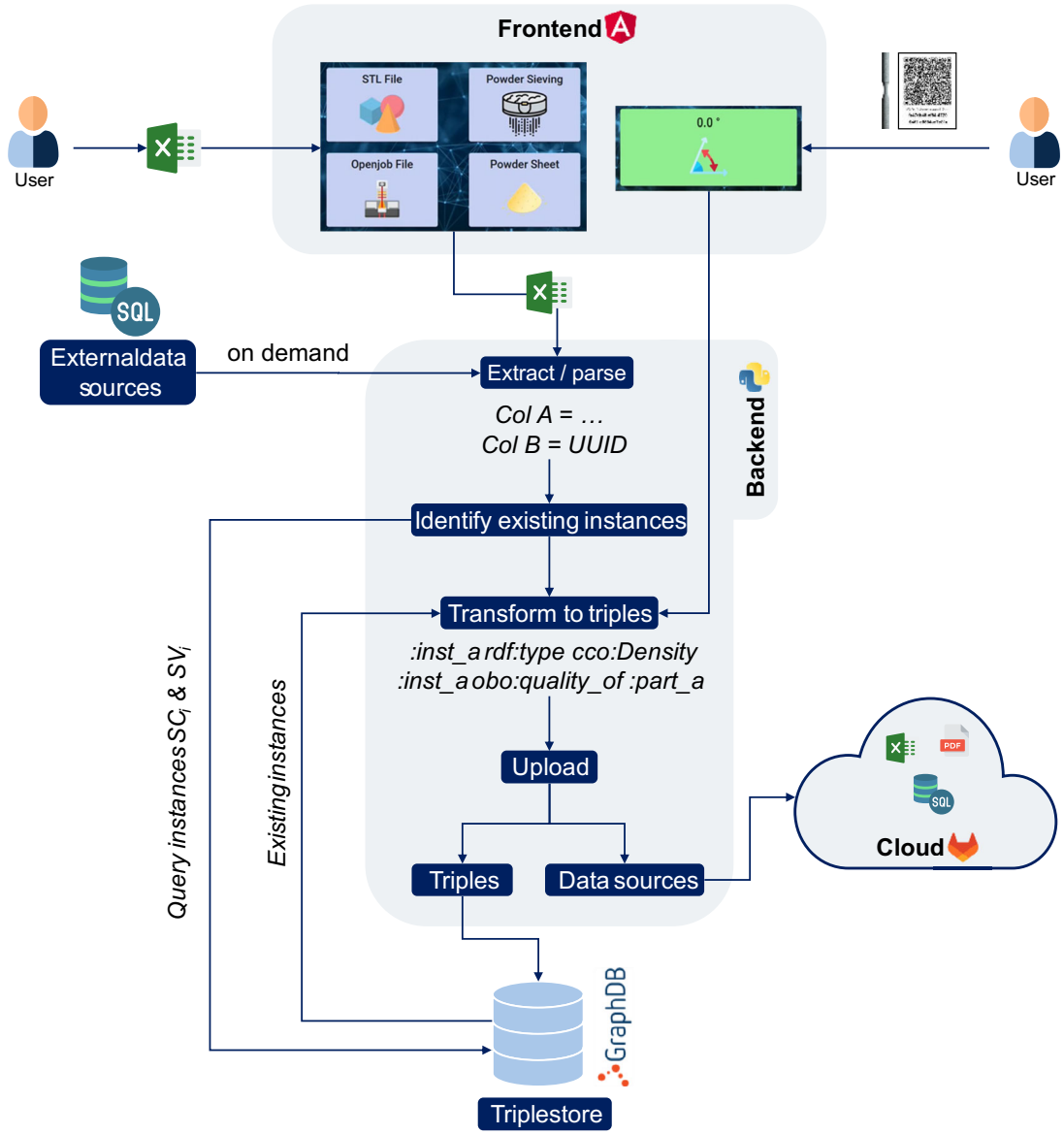

Figure 9. Architecture of linkage layer, implemented as a web application for enabling data import and data access.

The implemented and discussed import mechanisms include three main approaches: user-initiated uploads, user-inputted information, and external data access. User-initiated uploads allow users to upload files with a predefined structure, as captured in the modeling patterns blueprint. Examples include tabular files from density measurements or a build job file. User-inputted information involves users filling in missing information at predefined fields in the digital part record, as discussed in Section 4.5. External data access is enabled by accessing external data sources, such as an *SQL* database containing a part's current position tracked via an *ultrawide-band (UWB)* system, which are accessed on demand when data queries are made.

The import procedures can be segmented into *Extract, Identify, Transform, Load (EITL)*, adapted from traditional *Extract, Transform, Load (ETL)* approach for databases.

During the extraction phase, the data source is accessed, which may involve parsing a spreadsheet or establishing a connection to an *SQL* database. A key limitation at this stage is the requirement for a fixed data structure, such as the structure of the build job file. Increasing flexibility to accommodate build job files from different slicer versions directly is linked to coding and maintenance effort. Incorporating thousands of different file types, which are dynamically changing in industrial fields, requires more efficient strategies.

In the identification phase, existing instances in the triplestore referenced in the extracted data are queried using their sets of criteria for unique identification *SC* and *SV* to ensure flexibility in the upload sequence. For example, the powder instance used in a specific *PBF-LB/M* process might already exist with its charge number when the user uploads a build job file.

In the transformation stage, triples are generated according to the discussed modeling patterns, incorporating existing instances and creating new instances and data values linked with object and data properties. Historically, *Python's owlready2* library has been used to create instances and triples, but this resulted in duplicated triples when combined with an external triplestore.[49] A more efficient method used subsequently involves directly creating triples using Python string variables.

Finally, during the loading phase, a *SPARQL* insert command uploads the triples to the triplestore. User-uploaded files are managed through *GitLab* in the current demonstrator implementation, ensuring they are versionable and accessible for verification, while time-series data are added to an *SQL* database.

Figure 10 illustrates the triples to be created in transform step for one row of three columns of a spreadsheet.

Assuming one line of code per triple and excluding inverse relations, this small example requires nine lines of code for the pure formulation of the triples. Before a new instance is created with a triple like *:instance rdf:type :Class*, its existence must be verified in the triplestore to prevent duplication in identification step. Additionally, the type for both newly created and existing instances must be defined. Following the blueprint, containing hundreds of triples to be stated and requiring instances to be identified result in high coding effort. To streamline the coding process and improve maintainability, two approaches are employed. With first approach reoccurring modeling patterns are encapsulated in semantic templates. These are separate functions designed for either creating or querying triples for specific instances based on their *SC* and *SV*. For example, a function might search for a part based on a *UUID*, and if no existing part instance is found, it will automatically create the necessary triples for its definition. The second approach uses reasoning mechanisms to further simplify the process and ensure data integrity, which is discussed in Section 4.4.

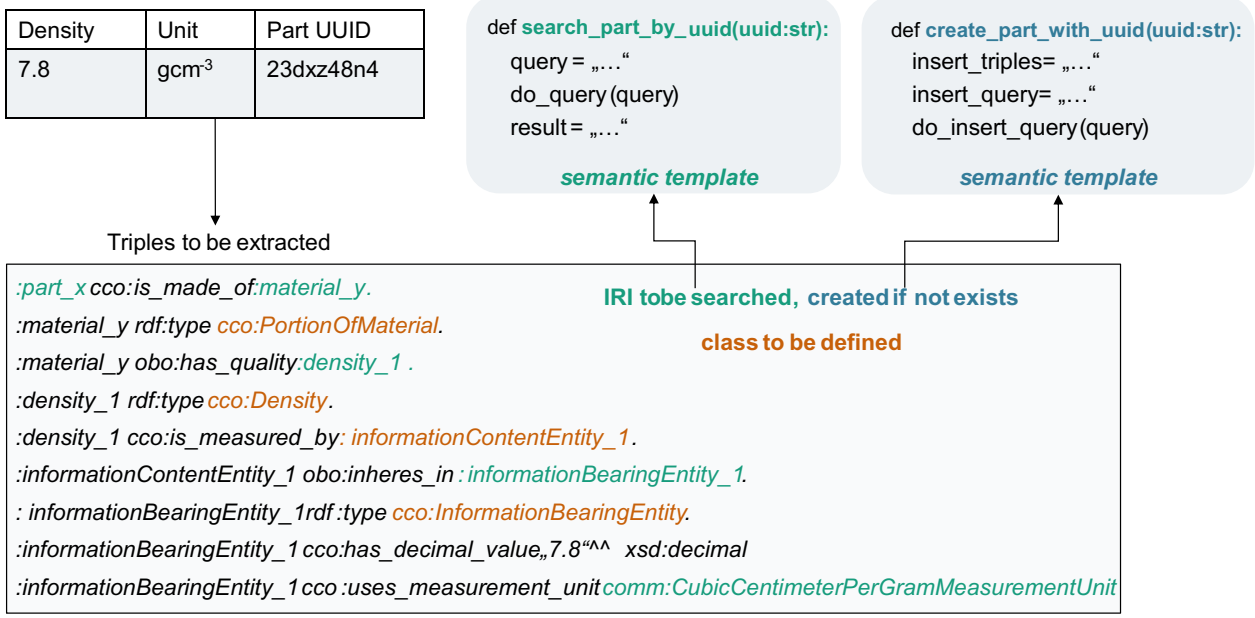

Figure 10. Triples to be created in transform step and instances to be searched in identification step for three columns in a spreadsheet.

4.4. Logic Engine and Reasoning

OWL reasoning utilizes *object properties* derived from defined rules and axioms. *OWL* includes predefined rules and axioms such as transitive, symmetric, and inverse properties, as well as class and property equivalence, extensively detailed in the language specification.[12]

Similar to selecting the appropriate data sources and data in Section 3.1, the purpose of each rule must be clearly defined. For the current application, rules are established to infer information for the digital part records and enhancing *reusability* by increasing the number of relationships defined. Depending on the complexity of the logics to be integrated, three methods are employed. The first method uses the integrated reasoner in *GraphDB* with an *OWL2-RL* profile along with standard *OWL* axioms.[50] This covers automatic inference, such as deriving inverse and symmetric properties, without the need to formulate explicit rules. The second method employs custom rules, which are tailored to meet specific application needs. The third approach utilizes a custom *Python*-based solution, where queried information is processed using *Python* scripts for inference and arithmetic operations. Examples of inferred statements from the three different reasoning approaches are shown in **Figure 11**.

The reasoner in *GraphDB* is used for declarative rules, requiring the creation of *properties* only. Predefined *OWL* axioms such as transitivity *("if :A :has_output :B, then :B :is_output_of :C")*, property hierarchies *("if :A :has_output :B, then :A :has_participant_at_all_times :B")*, and class hierarchy inference *("if :A rdf:type :Steel316L, then :A rdf:type :Steel")* reduce the amount of explicitly stating these triples. Using *BFO* and *CCO* axioms, logical inconsistencies are identifiable, for example if the pattern for a quality, the information about a quality, and the actual values are mishandled.

Domain-specific rules are incorporated via a PIE file, serving as the rule specification for *GraphDB*. These rules are primarily aimed at reclassifying existing instances, such as ensuring manufactured parts' material instances match the raw material used or that parts share the same orientation when utilizing the same geometry.

For more complex logic and arithmetic tasks, a *Python*-based approach is adopted. This involves executing *SPARQL* queries at low-second intervals to retrieve and process information, generate new triples, and upload them. For example all powder instances and their mass fractions of chemical elements are queried and triples depicting the material type are generated and uploaded.

Using reasoning to support the generation of *FAIR* data has been found to provide several key benefits. First, it enhances efficiency by significantly reducing the number of triples that need to be explicitly defined in the linkage layer, thereby decreasing implementation and maintenance efforts. In the current model, out of 1.1 million triples, only 100 000 are explicitly defined while one million are inferred, resulting in a 90% reduction in direct triple formulation. Additionally, reasoning minimizes the necessary user input by propagating information across multiple instances. For example, it can transfer the material type and orientation from a specific manufactured part to all parts meeting certain criteria. Furthermore, reasoning ensures logical consistency within the model by applying axioms from *BFO* and *CCO*, which help maintain the coherence of the system. Finally, it enhances the reusability of data by automatically adding properties that transform data into actionable information, thereby making the data more usable for future use.

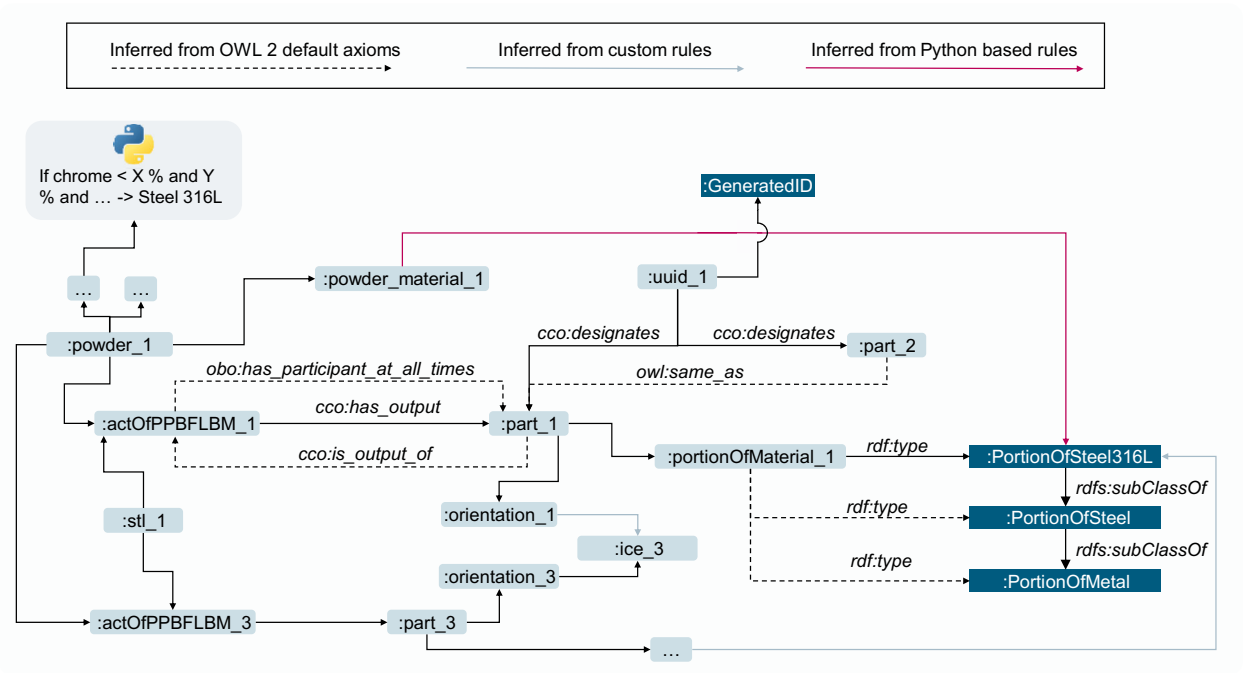

Figure 11. Exemplarily inferred statements from reasoning mechanisms for *OWL* default axioms, custom rules, and the *Python*-based approach.

The studies conducted in this research have been limited to approximately one million triples. As the volume of data typically encountered in semantic technologies can vastly exceed this number, the scalability of the approach when handling significantly larger datasets requires further investigation. Advancements in reasoning technologies, which represent a distinct research field, have led to substantial improvements, with tasks that previously took hours now being solvable in seconds due to enhanced reasoner performance.[51] Studies have shown that current methodologies are capable of managing reasoning processes efficiently even with billions of triples.[51] Future strategies to address the challenges posed by large-scale data might include the parallelization of reasoning tasks, a reduction in expressivity to streamline processing, or distributing the data in different partitions.

4.5. Query Layer and Use-Case Demonstration

Once data are structured in the proposed system, use cases derived from the competency questions in Table 1 simplify formulating and executing queries and setting up methods for visualizing results. For identifying material properties and selecting machine parameters, a dashboard that plots process–material relationships is implemented, based on the architecture depicted in **Figure 12**.

Based on expert interviews, processing factors crucial for part design and parameter selection are identified and integrated into query templates in the backend. When a user selects a material type and characteristic on the dashboard, the corresponding query template is populated and submitted to the triplestore. If the triplestore retrieves the requested properties directly, they are sent to frontend for visualization. For external data sources, such as *SQL* databases containing time series of thermal measurements and position information from the *UWB* system used in the current work, the triplestore provides the schema and access details necessary to query the properties from the external source.

Figure 13 displays one of several plots generated when a user selects aluminum and fatigue properties. It shows the number of cycles until breakage for different stress amplitudes and types of mechanical processing. From this information, upper and lower fatigue strength limits can be determined for part design, and turning might be selected as the mechanical process if the goal is to optimize fatigue strength. By deriving such instructions, information is transformed to knowledge according to *Ackoff*.[13]

For process parameter selection, the objective is to optimize a material property by selecting appropriate values for manipulatable factors along the processing chain. To facilitate this, material properties are plotted based on FAIR-captured information, such as postprocessing methods, build-platform coordinates, part orientations, part volumes, and machine parameters. For example, **Figure 14** shows a plot of steel density as a function of used laser power.

From the *FAIR* data currently available, it is suggested to set the laser power between 240 and 270 W for steel to minimize the probability of reduced density. The impact of this decision should be evaluated by analyzing additional material properties, such as impact strength, within this parameter range.

For each instance in the ontology that provides *UDPI*, being uniquely identifiable according to their *SC* and *SV*, the ontology automatically serves as the data source for a digital object record, containing information about the object's processing and qualities. In the current use case, this is manifested as a digital part record. The specific content and visualization of such a record are dictated by the use case requirements, such as the ability to trace a part's manufacturing history or meet regulatory requirements for information on manufactured goods.

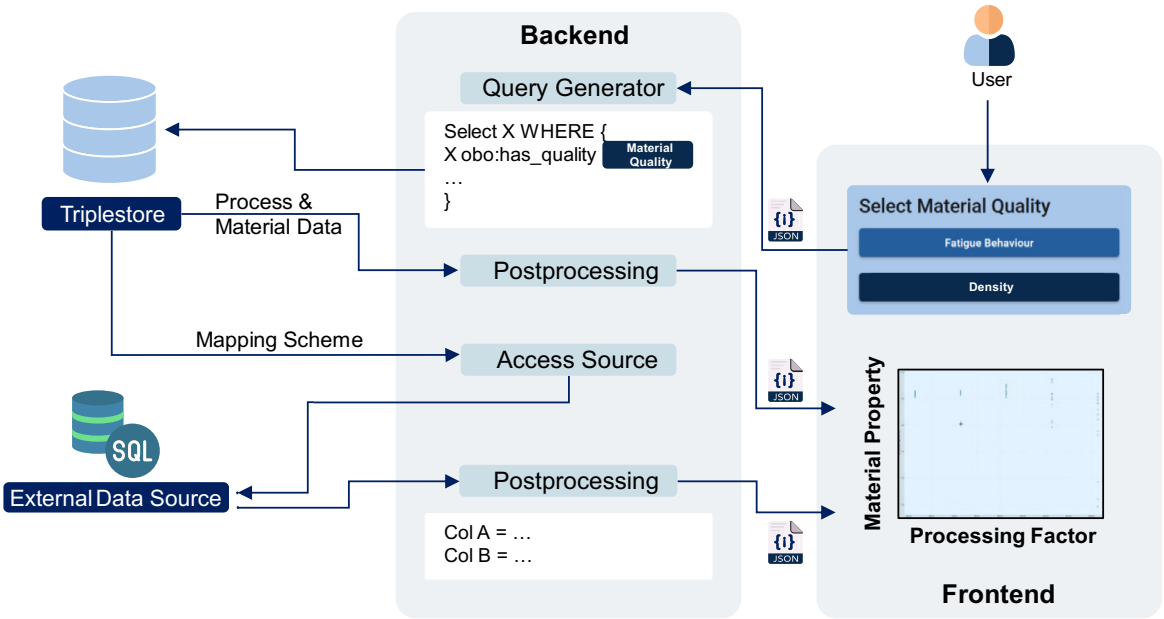

Figure 12. Main architecture and workflow in query layer.

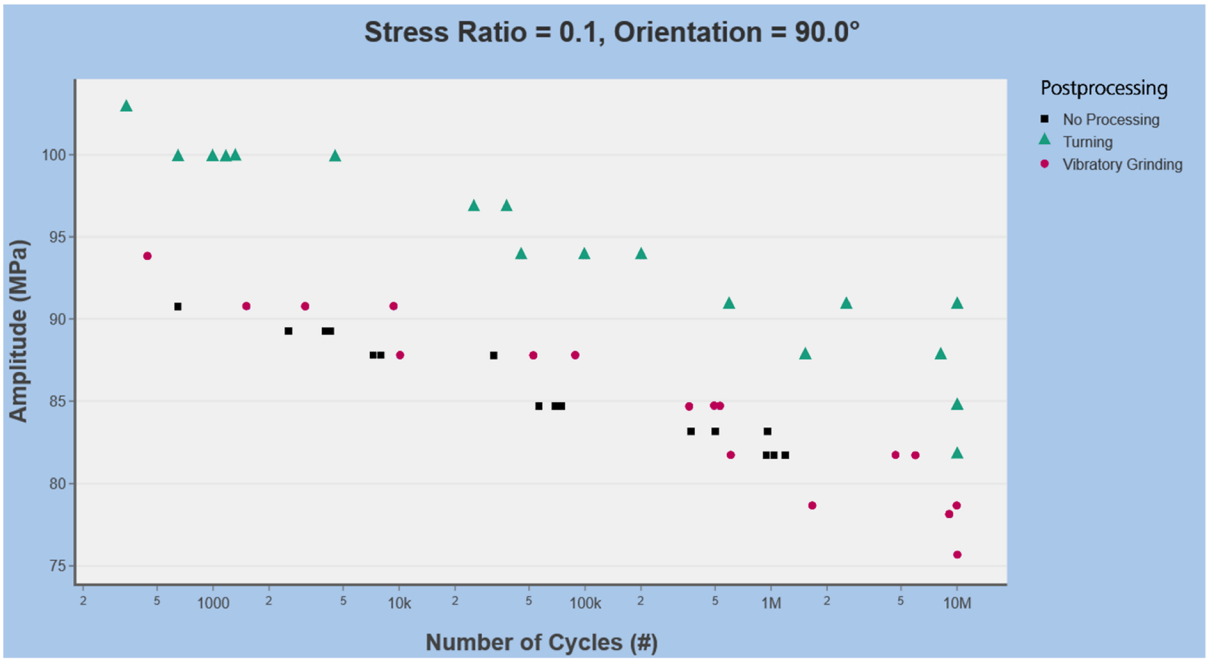

Figure 13. Exemplary generated plot showing stress amplitude versus the number of cycles until fracture for aluminum under different mechanical postprocessing conditions.

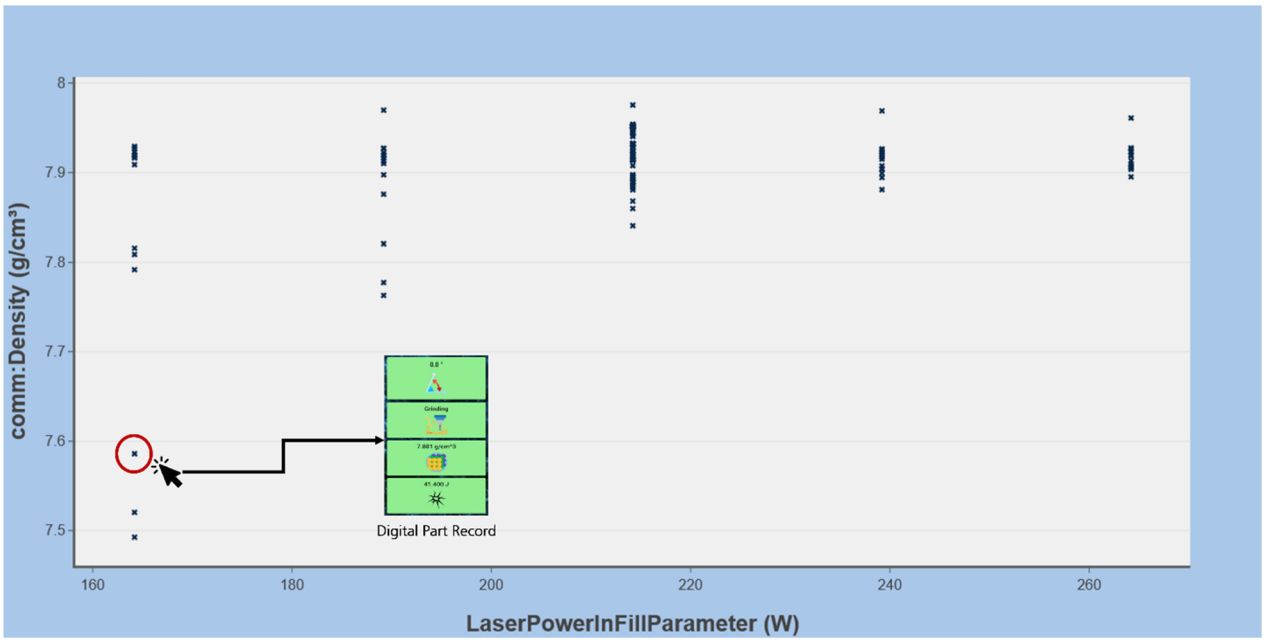

Figure 14. Exemplary generated plot for steel 316L with density over laser power parameter.

A snippet of the *digital part record* of a specimen as visualized in the frontend, showing its 3D geometry, powder charge number, build orientation, machine parameters, and its current location based on the *UWB* tracking, is given in **Figure 15**. Access to this record is facilitated by searching the part's *UUID* in the web app, scanning the *QR* code on the part, or clicking on a data point in the material process dashboard.

Identified conclusions from reusing *OWL*-based *FAIR* data include the following. 1) *Centralizing Data Storage:* Keeping all data values within a single triplestore, including time series data, eliminates the need to access external data sources, reducing the effort required to query information and potentially enhancing *FAIR* compliance. However, this is impractical for very large datasets like monitoring data, as it can degrade the performance

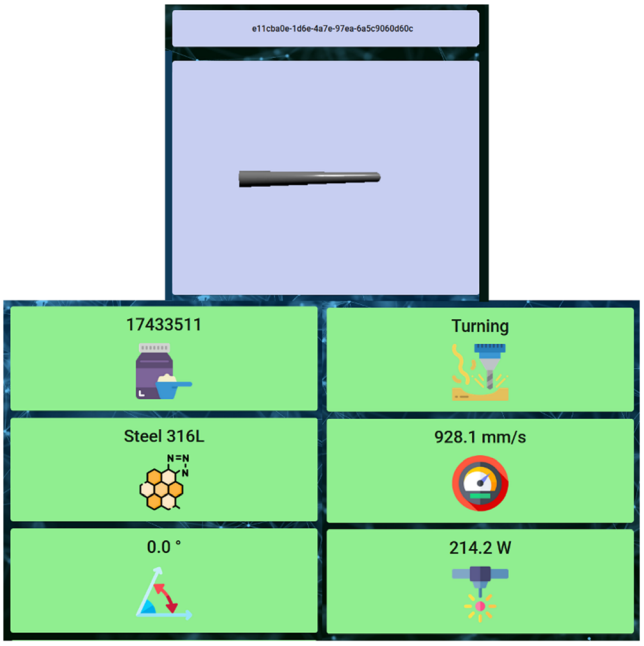

Figure 15. Exemplary excerpt from the digital part record, showcasing processing and material information for each part, adaptable for further objects along the process chain.

of the triplestore and significantly increase query execution times; 2) *Efficient Information Retrieval:* Retrieving required information for use cases in this project is reduced to formulate a query once data are structured *FAIR*. While this demands familiarity with the inherent modeling patterns, it is considered more economical and can enable the retrieval of information from different sources that was not manually possible before. Quantitative studies are needed to evaluate this benefit; 3) *Dynamic Knowledge Generation:* Generating knowledge by deriving instructions, such as selecting laser power based on specific parameters, relies on the set of currently available *FAIR* data. Over time, as more data points are added (e.g., an additional 1000 data points for density and laser speed), the derived instructions may change, underscoring the importance of having expansive *FAIR* data; 4) *Automation of instructions:* Currently, instructions are manually derived but could potentially be generated automatically using statistical methods applied to the structured data. This extends beyond the pure generation of *FAIR* data, suggesting that the *OWL* ontology itself could be modeled to encapsulate derived instructions; 5) *Taxonomy and data specificity:* The inherent taxonomy offers a mechanism to determine the generality of facts, providing labels for each data point. For example, it allows for specific conclusions like "the parabolic correlation between energy density and mass density is valid for steel, but not for all metals."

5. Summary and Conclusion

Based on three competency questions addressing part design, parameter selection, and part processing history aspects to consider, advantages and limitations of an *OWL*-based approach for *FAIR* data generation in the *PBF-LB/M*, representative for *AM*, have been identified.

UDPI of physical objects and instances representing them in the ontology is required for data assignment, providing a digital object record, extending *findability* for practical reuse of generated *FAIR* data in productive environments, and is the prerequisite for enabling a *digital twin*. It requires formulating *sets of criteria SC* and their *SV* for objects involved, such as powders, machines, and manufactured parts. Assigning a *UUID* to each part states one possibility and requires its physical attachment and instantiation in the ontology, exemplarily discussed using *QR* codes.

Interoperability implies the reuse of top- and midlevel ontologies for shareable vocabularies. The approach used with glossaries and graphical diagrams lacks efficiency due to dupl icated entities, lacking synchronization methods, and the complexity of discussions with domain experts. Hurdles for practical work with *BFO* and *CCOs* include the necessary expertise to understand modeling philosophy, ambiguities in conforming modeling, and potential overhead due to rigid modeling patterns. In addition to enhanced *findability*, advantages from practical work include domain expandability along the processing chains, differentiation between specifications and real occurrences, differentiated name levels for objects, sequence modeling, differentiation between qualities and information about them, and data source allocation.

Data integration is discussed using *Python* modules for parsing structurally predefined exchange formats. Drawbacks include inflexibility in data format changes and high coding efforts for triple formulation and querying existing instances. Semantic templates incorporating repeatedly occurring modeling patterns for triple generation and querying reduce the coding and maintenance effort. Reasoning reduces the effort of pure triple formulation by 90% by inferring transitive and inverse relations and allows the transition of object information to a higher level of generality, reducing necessary user input and enhancing reusability. More complex inferences, such as material type classifications, require an externalized reasoning approach, discussed using *Python*.

Reuse of *FAIR* data was demonstrated based on a material data dashboard. Plotting the processing and material characteristics allows the selection of processing windows, whereas the taxonomic labeling enables conclusions about the generality of visually found correlations to material types. Keeping all data values in the triplestore itself enhances accessibility but reduces performance, whereas time series data are externalized. Retrieval of required information is assumed to be more efficient, as the step of combining heterogeneous data sources is carried out once at the step of creating the *FAIR* data. Generating knowledge by deriving instructions, for example, for parameter selections, might change if the underlying amount of *FAIR* data changes, motivating their generation in the long term and addressing more automated ways for instruction identification. For each object enabled with *UDPI*, a *digital object record* can be provided, whereas the information to be contained must be specified by use case or legally. The demonstrated digital part record, carrying information about the processing history, showed the capability of *OWL*-based *FAIR* data in *AM* to be reused. Adapting the procedure for further objects along the process chain, such as

raw materials and product structures, yields the potential to create a connected source of digital twins.

6. Future Work

Transitioning the results to production readiness involves several tasks. Standards for criteria *SC* and *SV* must be defined on different levels, that is, company wide and at the level of manufacturer alliances. Methods to physically attach and digitally apply *UUIDs* must be further automated, for example, using printed labels, for faster processing and reducing error proneness. *UDPI* must be successively expanded to objects involved in the process chain, like powders and machines, to further ease data allocation and enhance reusability. Using *BFO* and *CCO* has shown to be capable of harmonizing different data sources into a consistent model. Further applicability may be achieved by applying it to different domains and harmonizing the modeling patterns as done in the material science domain in the *MaterialDigital* initiative. For scalable and more efficient data source integration, methods dealing with changing and new data formats like supervised large language model-based approaches need to be researched and integrated, incorporating data source versioning and synchronization methods. A standardized, iterative, versionable method for ontology-based *FAIR* data generation is required. Currently, manually identified correlations and root cause analysis need to be automated to further ease the process of identifying root cause correlations, for example, by incorporating statistical approaches and applying graph neural networks, while the structured material and process information in the ontology provides a comprehensive basis to be traversed. Using the defined sets of criteria for unique identification, further legal standards that dictate the information to be provided by product manufacturers might drive digital object record development and require standardized authorization and authentication methods.

Acknowledgements

The authors thank the *German Federal Ministry of Education and Research (BMBF)* for its financial support in the project *ODE_AM* through project funding *FKZ 13XP5117B*. Additionally, the authors thank *Clemens Gonnermann* and *Maximilian Holland* from *Fraunhofer IGCV* for their revisions and *Max Horn* from *Fraunhofer IGCV*, *Heiko Beinersdorf* from *MFPA Weimar*, *Mohamed Kamal* from *TU Hamburg*, and *Jan Reiman* from *TU Ilmenau* for project support.

Open Access funding enabled and organized by Projekt DEAL.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Thomas Bjarsch: conceptualization (lead); data curation (lead); methodology (lead); resources (lead); writing—original draft (lead); writing—review & editing (lead). **Klaus Drechsler**: methodology (equal); supervision (supporting); writing—review & editing (supporting). **Johannes Schilp**: conceptualization (supporting); supervision (supporting); writing—review & editing (supporting).

Data Availability Statement

The created ontologies are publicly available at https://w3id.org/ODE_AM/.

Keywords

additive manufacturing, basic formal ontologies, findable, accessible, interoperable, reusable, ontologies, powder bed fusions

Received: June 27, 2024
Revised: October 21, 2024
Published online: February 7, 2025

[1] S. Li, K. S. Aggour, Y. Lu, P. Coutts, B. Harris, A. Kitt, A. Lupulescu, M. McNair, L. Mohr, M. Vasquez, *Advanced Manufacturing Series* **2023**, National Institute of Standards and Technology, Gaithersburg, MD, pp. 500–501.
[2] K. S. Aggour, V. S. Kumar, P. Cuddihy, J. W. Williams, V. Gupta, L. Dial, T. Hanlon, J. Gambone, J. Vinciquerra, in *IEEE Int. Conf. on Big Data*, Los Angeles, CA **2019**.
[3] W. E. Frazier, Y. Lu, P. Witherell, A. Kitt, *Additive Manufacturing Design and Applications*, ASM International, Almere, FL **2023**, p. 2.
[4] W. E. Frazier, Y. Lu, P. Witherell, R. Fryan, A. Kitt, *AM&P Tech. Art.* **2021**, *179*, 12.
[5] M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, et al., *Sci. Data*, **2016**, *3*, 160018.
[6] P. Borst, *Construction of Engineering Ontologies for Knowledge Sharing and Reuse*, Centre for Telematics and Information Technology, Enschede **1997**.
[7] T. R. Gruber, *Knowl. Acquis.* **1993**, *5*, 199.
[8] R. Studer, V. R. Benjamins, D. Fensel, *Data Knowl. Eng.* **1998**, *25*, 161.
[9] M. R. Genesereth, N. J. Nilsson, *J. Symb. Logic* **2014**, *55*, 1304.
[10] S. Staab, R. Studer, *Handbook on Ontologies*, Springer, Berlin **2009**.
[11] R. Arp, B. Smith, A. D. Spear, *Building Ontologies with Basic Formal Ontology*, The MIT Press, Cambridge, MA **2015**.
[12] World Wide Web Consortium (W3C), *OWL 2 structural specification and functional-style syntax*, https://www.w3.org/TR/2012/REC-owl2-syntax-20121211/ (accessed: May 2024).
[13] R. L. Ackoff, *J. Appl. Syst. Anal.* **1989**, *16*, 3.
[14] P. Grimmel, J. Wessel, M. Mennenga, C. Herrmann, *SSRN J.* **2022**, *5*, 4073026.
[15] J. N. Otte, J. Beverley, A. Ruttenberg, *Appl. Ontol.* **2022**, *17*, 17.
[16] Platform MaterialDigital, *Initiative for digitalization of materials*, https://www.materialdigital.de/ (accessed: May 2024).
[17] M. Jensen, D. Glimbaugh, *Repository holding the current released version of the Common Core Ontology suite*, https://github.com/CommonCoreOntology/CommonCoreOntologies, (accessed: May 2024).
[18] R. Rudnicki, *Modeling Information with the Common Core Ontologies*, https://www.nist.gov/system/files/documents/2021/10/14/nist-ai-rfi-cubrc_inc_003.pdf (accessed: August 2024).
[19] DOI org, *Non-profit organization to provide digital object identifiers*, https://www.doi.org/ (accessed: October 2024).
[20] HITS gGmbH, *Consortium of services for research data management*, https://fair-dom.org/ (accessed: October 2024).

[21] Leibniz Information Centre for Economics, *Go FAIR - Webiste for supporting fair data creation*, https://www.go-fair.org/ (accessed: October 2024).
[22] M. Mohd Ali, R. Rai, J. N. Otte, B. Smith, *Comput. Ind.* **2019**, *105*, 191.
[23] C. Liu, L. Le Roux, C. Körner, O. Tabaste, F. Lacan, S. Bigot, *J. Manuf. Syst.* **2022**, *62*, 857.
[24] A. Kuan, K. S. Aggour, S. Li, Y. Lu, L. Mohr, A. Kitt, H. Macdonald, *Integr. Mater. Manuf. Innovation* **2024**, *13*, 105.
[25] S. Li, S. Feng, A. Kuan, Y. Lu, *JOM* **2024**, *76*, 1905.
[26] R. Wang, C. F. Cheung, *Comput. Ind.* **2023**, *149*, 103912.
[27] B.-M. Roh, S. R. T. Kumara, H. Yang, T. W. Simpson, P. Witherell, A. T. Jones, Y. Lu, *J. Comput. Inf. Sci. Eng.* **2022**, *22*, 4055853 .
[28] B.-M. Roh, S. R. T. Kumara, H. Yang, T. W. Simpson, P. Witherell, Y. Lu, in *Proc. of the ASME Design Engineering Technical Conf.*, Virtual/Online **2021**.
[29] G. Formentini, C. Favi, M. Mandolini, M. Germani, *Proc. Des. Soc.* **2022**, *2*, 1371.
[30] A. Haruna, M. Yang, P. Jiang, *Proc. Inst. Mech. Eng.* **2022**, *237*, 1405.
[31] M. Mayerhofer, W. Lepuschitz, T. Hoebert, M. Merdan, M. Schwentenwein, T. I. Strasser, *IEEE Open J. Ind. Electron. Soc.* **2021**, *2*, 207.
[32] H. Ko, P. Witherell, Y. Lu, S. Kim, D. W. Rosen, *Addit. Manuf.* **2021**, *3*, 101620.
[33] J. Han, D. Schaefer, *Proc. CIRP* **2019**, *81*, 850.
[34] Q. Jarrar, F. Belkadi, R. Blanc, K. Kestaneci, A. Bernard, *Int. J. Prod. Res.* **2022**, *61*.
[35] Q. Jarrar, F. Belkadi, A. Bernard, *IFIP Advances In Information and Communication Technology, Springer International Publishing, New York City*, NY **2021**.
[36] M. Huang, B. Fan, L. Chen, Y. Pan, Y. Qin, *Appl. Sci.* **2023**, *13*, 2926.
[37] M. Huang, N. Zheng, Y. Qin, Z. Tang, H. Zhang, B. Fan, L. Qin, *Processes* **2022**, *10*, 1290.
[38] Y. N. Kulchin, V. V. Gribova, V. A. Timchenko, M. V. Polonik, D. S. Pivovarov, D. S. Yatsko, P. A. Nikiforov, A. I. Nikitin, *Lecture Notes on Data Engineering and Communications Technologies, Springer, Cham, CH* **2022**.
[39] L. Guo, Y. Cheng, Y. Zhang, Y. Liu, C. Wan, J. Liang, in *IEEE Int. Conf. on Industrial Informatics (INDIN), Piscataway, NJ* **2021**.
[40] I. Rojek, D. Mikołajewski, P. Kotlarz, M. Macko, J. Kopowski, *Bull. Polish Acad. Sci.* **2021**, *69*, 136722.
[41] V. S. Kumar, P. Cuddihy, K. S. Aggour, in *Proc. of the ACM SIGMOD Int. Conf. on Management of Data*, Amsterdam, NL **2019**.
[42] M. Uschold, M. Gruninger, *Knowl. Eng. Rev.* **1996**, *11*, 93.
[43] International Organization for Standardization, *American Society for Testing and Materials, DIN EN ISO/ASTM 52900:2022-03*, Beuth Verlag GmbH, Berlin **2022**.
[44] W. Kritzinger, M. Karner, T. Georg, J. Henjes, W. Sihn, *IFAC-PapersOnLine* **2018**, *51*, 1016.
[45] A. Ruttenberg, *Basic Formal Ontology (BFO)*, https://basic-formal-ontology.org/ (accessed: May 2024).
[46] JGraph Ltd, *Tool to collaboratively create diagrams*, https://www.drawio.com/ (accessed: May 2024).
[47] BFO Discuss, *Discussion Group for BFO modeling*, https://groups.google.com/g/bfo-discuss (accessed: May 2024).
[48] S. Chávez-Feria, R. García-Castro, M. Poveda-Villalón, *Tool to transform an ontology diagram into OWL code*, https://github.com/oeg-upm/Chowlk (accessed: May 2024).
[49] J. B. Lamy, *owlready2 documentation*, https://owlready2.readthedocs.io/en/latest/ (accessed: May 2024).
[50] Ontotext, *GraphDB triplestore*, https://graphdb.ontotext.com/ (accessed: June 2024).
[51] M. Ruster, *Large-Scale Reasoning with OWL*, http://arxiv.org/pdf/1602.04473v1 (accessed: October 2024).
[52] Fraunhofer-Institut für Gießerei-, Composite- und Verarbeitungstechnik IGCV, Metallbasierte Additive Fertigung, https://www.igcv.fraunhofer.de/de/forschung/kompetenzen/additive_fertigung_am/am_metall.html (accessed: June 2024).
[53] C. Grandvallet, F. Pourroy, F. Vignat, *Lecture Notes in Mechanical Engineering*, Springer, Cham, CH **2023**.
[54] M. Bonello, P. Farrugia, *Comput. Aided Des. Appl.* **2023**, *20*, 170.
[55] M. Gmeiner, W. Lepuschitz, M. Merdan, M. Lackner, *Lecture Notes in Networks and Systems*, SAGE Publications, Los Angeles, CA **2022**.
[56] J. Ahn, J. Doh, S. Kim, S.-I. Park, *Micromachines* **2022**, *13*, 1672.
[57] R. Chen, Y. Lu, P. Witherell, T. W. Simpson, S. Kumara, H. Yang, *IEEE Rob. Autom. Lett.* **2021**, *6*, 6032.
[58] B.-M. Roh, S. R. T. Kumara, P. Witherell, T. W. Simpson, *J. Mater. Eng. Perform.* **2021**, *30*, 8784.
[59] F. Belkadi, E. M. Sanfilippo, A. Bernard, L. M. Vidal, *Comput. Aided Des. Appl.* **2020**, *17*, 1278.
[60] W. Lepuschitz, T. Trautner, M. Mayerhofer, S. M. Fallah, I. Ayatollahi, M. Merdan, in *IECON Proc. (Industrial Electronics Conf.)*, Lisbon, PT **2019**.
[61] C. Qian, Y. Zhang, Y. Liu, Z. Wang, *J. Cleaner Prod.* **2019**, *241*, 118379.
[62] Y. Xiong, A. G. Dharmawan, Y. Tang, G. S. Soh, D. W. Rosen, in *Solid Freeform Fabrication 2019: Proc. of the 30th Annual Int. Solid Freeform Fabrication Symp. - An Additive Manufacturing Conf.*, Austin, TX **2019**.
[63] S. Kim, D. W. Rosen, P. Witherell, H. Ko, *J. Comput. Inf. Sci. Eng.* **2019**, *19*, 041014.
[64] T. Moges, P. Witherell, G. Ameta, in *ASME Int. Mechanical Engineering Congress and Exposition, Proc. (IMECE)*, Salt Lake City, UT **2019**.