Progress in Additive Manufacturing
https://doi.org/10.1007/s40964-024-00660-7

FULL RESEARCH ARTICLE

Design and implementation of a machine log for PBF-LB/M on basis of IoT communication architectures and an ETL pipeline

Konstantin Poka[1] · Sozol Ali[1] · Waleed Saeed[1] · Benjamin Merz[1,2] · Martin Epperlein[1] · Kai Hilgenberg[1]

Received: 2 May 2024 / Accepted: 7 May 2024

Abstract
Powder Bed Fusion with Laser Beam of Metals (PBF-LB/M) has gained more industrial relevance and already demonstrated applications at a small series scale. However, its widespread adoption in various use cases faces challenges due to the absence of interfaces to established Manufacturing Execution Systems (MES) that support customers in the predominantly data-driven quality assurance. Current state-of-the-art PBF-LB/M machines utilize communication architectures, such as OPC Unified Architecture (OPC UA), Message Queuing Telemetry Transport (MQTT) and Representational State Transfer Application Programming Interface (REST API). In the context of the Reference Architecture Model Industry 4.0 (RAMI 4.0) and the Internet of Things (IoT), the assets, particularly the physical PBF-LB/M machines, already have an integration layer implemented to communicate data such as process states or sensor values. Missing is an MES component acting as a communication and information layer. To address this gap, the proposed Extract Transform Load (ETL) pipeline aims to extract relevant data from the fabrication of each build cycle down to the level of scan vectors and additionally to register process signals. The suggested data schema for archiving each build cycle adheres to all terms defined by ISO/TC 261—Additive Manufacturing (AM). In relation to the measurement frequency, all data are reorganized into entities, such as the AM machine, build cycle, part, layer, and scan vector. These scan vectors are stored in a runtime-independent format, including all metadata, to be valid and traceable. The resulting machine log represents a comprehensive documentation of each build cycle, enabling data-driven quality assurance at process level.

Keywords Data-driven quality assurance · Laser powder bed fusion · FAIR data

1 Introduction and motivation

Upon entering the shop floors of production facilities, new requirements emerge in terms of process control and data acquisition for PBF-LB/M [1]. To integrate AM machines into value chains of production, they should be interoperable with the Enterprise Resource Planning (ERP) systems [2]. This interoperability is crucial to manage orders. The MES distributes them across the fleet to the specific machine and obtains the data of the fabrication as feedback. In the MES, an entity of a build cycle should, therefore, get enriched with a set of all relevant data of the fabrication in order to be able to apply statistical process control (SPC) [3]. Such a dataset as the output of an ETL pipeline would be a traceable record of each build cycle for PBF-LB/M. The objective of this work is to generate it in a form of a PBF-LB/M machine log containing the instructions for the AM machine together with an image of the specific machine including all involved hard- and software components of the fabrication. External monitoring systems are not in the actual scope. Nevertheless, the process signals from the AM machine during fabrication could already generate a holistic view at process level. For that, the time series data published via an IoT protocol of the AM machine need to be registered on the fabrication instructions. This registration manifests a correlation between the timestamp, the sensor value, and the process entities, e.g., part and layer. By tracking the conditions of each layer and sensor changes during its fabrication, an easy root cause analysis is enabled. This analysis can then be

✉ Konstantin Poka
konstantin.poka@bam.de

[1] 9.6 Additive manufacturing of metallic components, Bundesanstalt für Materialforschung und –prüfung (BAM), Unter Den Eichen 87, 12205 Berlin, Germany

[2] Institute of Machine Tools and Factory Management, Technische Universität Berlin, Pascalstraße 8-9, 10587 Berlin, Germany

Published online: 21 May 2024

further supported by machine learning algorithms, due to the high data quality of the proposed machine log. Once the data is structured and mapped, the feature extraction is considerably simplified [4]. To achieve this, the following two questions are answered from the view of PBF-LB/M:

- How to retrieve the fabrication instructions and applied parameters runtime independent?
- What are the limitations of the proposed ETL pipeline and the overall setup?

2 Applied data sources

In the initial implementation, only process signals that are natively supported by the basic configuration of the AM machine are considered. Monitoring equipment, such as Powder Bed Cameras, Melt Pool Monitoring (MPM), or Optical Tomography (OT), is not incorporated. Nevertheless, the requirement to integrate these sources in future is considered during the design of the ETL pipeline. For the creation of the PBF-LB/M machine log, the data is organized and, if suitable, registered within the Machine Coordinate System (MCS). In any case, all process signals are synchronized to a common time scale in the format "yyyy-MM-dd HH:mm:ss.SSS" within the ETL pipeline. Subsequently, the smallest entity, which is used for data registration, is a scan vector. It is provided with a timestamp as well as start and end coordinates in the MCS. On the premise of an identical system time of the measurement device and a transformation rule on the MCS, the registration of external signals on the level of a hundredth of a second can therefore be enabled. To achieve this, the depth of information, mainly influenced by the actors equipped with interfaces and sensors, as well as the measurement frequency supported by each data point, needs to be analyzed first [5]. The utilized multi-laser AM machine is an EOS M 300–4.

Its structure can be described abstractly by the automation pyramid in Fig. 1. At the field level, all drives of the actors, of the z-axis of the build platform, of the recoater, of the feed system and of the compressor of the filtration system, transmit their current operating mode and additional information, such as torque, position, and rotations per minute (rpm), as a message. These messages, including a timestamp, are sent to the Programmable Logic Control

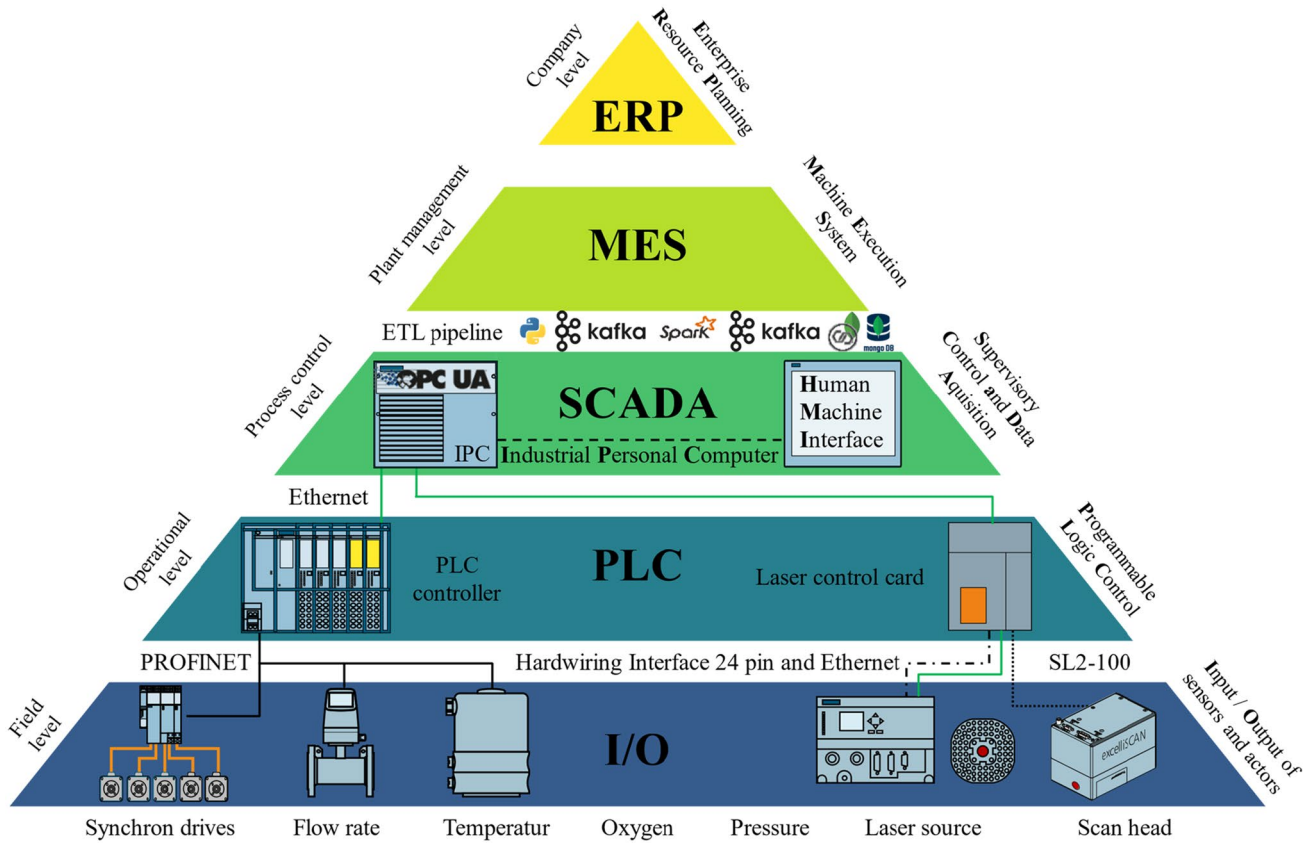

Fig. 1 PBF-LB/M machine and Extract Transform Load (ETL) pipeline within the automation pyramid

(PLC) controller at the operational level via a Binary Unit System (BUS) [6]. This communication model is in addition deployed for all sensors to establish a process control based on their signal values. The so-captured data can be categorized into the physical locations of the filtration system, the build chamber, and the environment. For the filtration system, the measurands include the oxygen concentration, the flow rate of the shielding gas, and the pressure difference of the in- and outlet. In the build chamber, the oxygen concentration of the atmosphere, the pressure, the humidity, and the temperature can be retrieved, along with the temperature of the build platform. To also monitor external influences, humidity and temperature at the installation site of the AM machine are tracked. The core component of each PBF-LB/M machine, the laser unit, is an exception to this seamless integration into the machine network. The assembly of a laser unit comprises a laser source, a control card, and a scan head.

The counterpart to the PLC controller at the operational level is the laser control card. Unlike the PLC, it does not support an open industrial Ethernet standard such as Process Field Network (PROFINET) in this AM machine, see Fig. 1. The tilting commands for the mirrors of the scan heads are transmitted from the laser control card via a proprietary SL2-100 protocol. Simultaneously, the modulation combined with the output power of the beams, is transferred to the laser sources. The execution of the scan vectors, regulated by these controls and adjusted process parameters, such as scan speed and laser power, takes place at a repetition rate of up to tens of kHz [7]. Despite the proprietary protocol, some data selected by the Original Equipment Manufacturer (OEM) can be accessed by the process control level through the Industrial Personal Computer (IPC). For each laser source, the operation time since the Installation Qualification (IQ) of ASTM 52930 [8], the rated laser power from the last calibration, and the current temperature are logged. However, the data of the scan vectors are not recorded.

A retrofit for the extraction of this data is not feasible as the laser control card lacks an interface and cannot be replaced without significant intervention in the machine network in the current setup. A workaround is established in Sect. 2.2, to at least obtain the data of the fabrication by an external source. The file of the build cycle, storing the instructions of the fabrication, is parsed assuming an ideal process with no additional delays. All other internally available datums collected by the PLC and the laser control card are aggregated by the IPC executing the Supervisory Control and Data Acquisition (SCADA) system, visualized in Fig. 1. At the level of process control also, an interaction via the Human Machine Interface (HMI) is implemented to set for instance the recoater speed and mode. All settings forwarded through the HMI, if supported, and the datums of the field level are published via the installed IoT communication architecture hosted by the IPC. The so-transferred data serves as the input for the ETL pipeline, which registers the data on process entities, such as part and layer. By adopting a standardized integration layer of RAMI 4.0, functionality across platforms is guaranteed [9].

2.1 Deployed IoT communication architecture

The PBF-LB/M machine of this work supports three IoT communication architectures, each with distinct scopes and applications [10]. The first is the OPC UA with its own standard IEC 62541 [11], overseen by the OPC foundation. It promotes device interoperability by establishing standard APIs rather than proprietary ones, and categorizes system software into clients and servers, with servers often located on devices like the PBF-LB/M machine IPC [12]. The OPC UA device model ensures semantic interoperability, defining generic object APIs even in AM contexts, as specified in the working draft of OPC UA 40540 [13]. The authentication of the client is strengthened in this setup through self-signed certificates and the Secure Hash Algorithm 256-bit (SHA256), ensuring secure communication. Entities, like devices and servers, can generate their own certificates, with SHA256 maintaining their integrity. This procedure adds an additional layer of security by verifying the integrity of the certificates and confirming their authenticity. This forms a robust foundation for secure OPC UA communication [14].

The second implemented architecture is MQTT. It is widely used in IoT, facilitates lightweight and efficient device communication through a publish/subscribe model, enhancing scalability. Known for its simplicity, MQTT is suitable for resource-constrained environments, excelling in high-throughput scenarios. Notably, MQTT operates without the need for a permanent connection, contributing to its flexibility and adaptability. Authentication in MQTT involves username–password pairs or advanced mechanisms like Transport Layer Security (TLS), crucial for securing data transmission. However, MQTT is not considered further, as the PBF-LB/M machine on this testbed supports only a limited number of data points and its spread in industry is lower [10].

The third supported architecture is a REST API as a specific implementation of the REST architecture tailored for a particular system, as observed on the IPC of the applied PBF-LB/M machine. When aligned with REST principles, it transforms into a RESTful API, delineating accessible resources through REST. While REST does not prescribe a specific technology, it is commonly implemented using the Hyper Text Transfer Protocol (HTTP). Resources are

identified by unique Uniform Resource Identifiers (URI), like the structure of web page addresses [15]. On the IPC of the PBF-LB/M machine, Swagger, a tool for API documentation, is deployed to enhance and simplify testing and exploration. API keys, serving as unique identifiers, are used for user authentication. Importantly, custom logics can be implemented for transferring process data, extending functionalities beyond what is covered by the OPC UA protocol in its standard usage, such as MPM or OT datasets.

In the following, the two remaining architectures, OPC UA and REST API, are evaluated for their suitability in the ETL pipeline, using the recoater torque machine datum as an example. The evaluation focuses on request speed and response rate, as well as on the robustness of the traffic. Therefore, the corresponding OPC UA node ns = 4; s = EOS. Machine.Recoater.AxisTorque is monitored via polling to retrieve the value every 0.1 s.

Additionally, the value is queried via the REST API using the GET request with curl: -X GET.

"https://<SI>/api/v6/recoater/axisTorque/current".

Both data queries are conducted separately for one hour each. The results are visualized in Fig. 2. It can be inferred that the OPC UA Client, implemented via the Python library opcua-asyncio, performs more efficiently in handling the request load and resulting traffic. In an ideal scenario, the client would retrieve 36,000 values per hour from the OPC UA server.

However, due to latency, the overall number of consumed values is 4,322 lower. This corresponds to a mean delay of 0.014 s for each transaction of the machine datum published via the OPC UA server. Opting for the OPC UA server as the source for machine data allows a more detailed examination of the latency between the creation of the sensor value and its retrieval. This latency is indicated by the differences between the source timestamp, which is assigned by the dedicated component of the PBF-LB/M machine during sensor value creation, the server timestamp when it is published, and the timestamp when the value is retrieved at the client. To ascertain this crucial parameter, pivotal for drawing conclusions regarding real-time capability, the torque sensor undergoes further interrogation. However, this investigation is conducted in subscription mode to mitigate the load in subsequent implementations in the ETL pipeline [10]. In this operating mode, a new value is only published when it surpasses a threshold represented by the dead band value. The value is then disseminated with a frequency at maximum equal to the revised sampling rate. In the case of the torque sensor, the sensor exhibits a measured value flicker above the dead band value, thus generating artificial traffic. The subscription to this allows a comparison of the source timestamp, server timestamp, and client timestamp during the retrieval process presented in Fig. 3. This evaluation is only valid under the premise that the server and the client are using the identical time server. The mean delay in the internal network of the machine is 200 ms. The average delay between the published value and the retrieving of the client is 900 ms. It can be assumed that the sensor values can

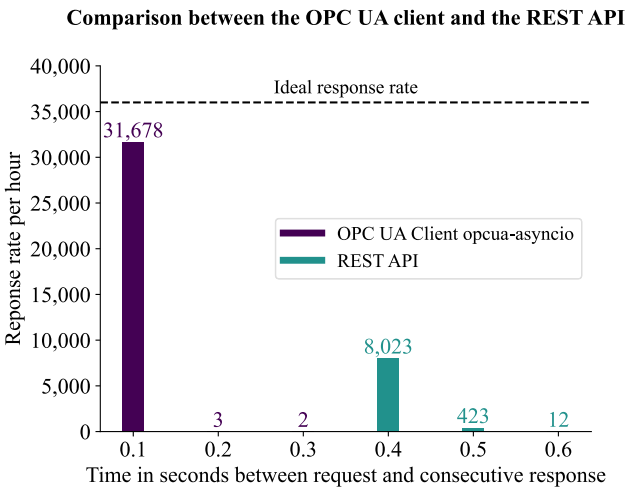

Fig. 2 Comparison of response speed of IoT protocols

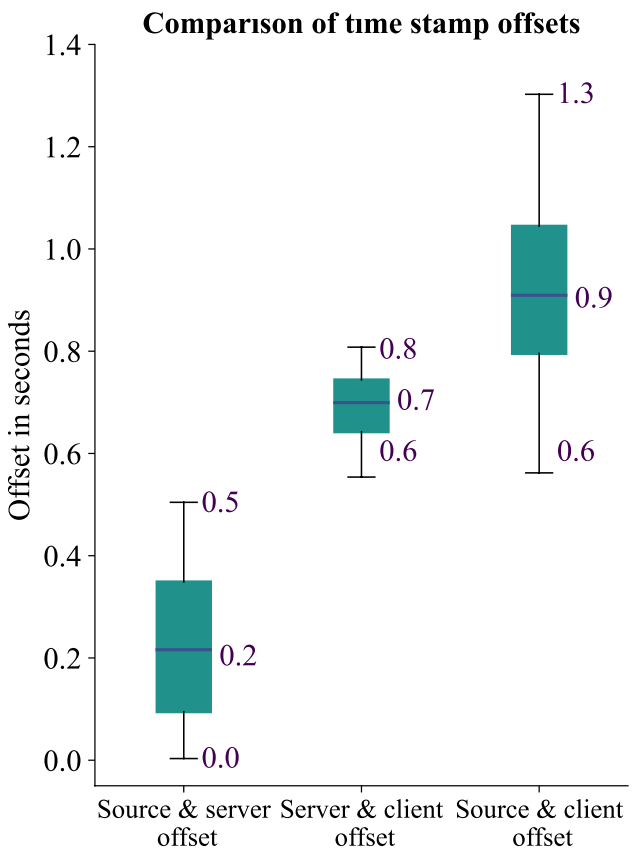

Fig. 3 Evaluation of the real time capability

be accurately assigned to build cycle entities, such as layer and part, provided that the revised sampling rate permits this. Despite the noted delay, this method is even suitable for rudimentary closed-loop control. For documentation purposes, the source timestamp of each machine datum is utilized because it is as precise as the integration time of the individual sensor.

For the execution of the individual scan vector, no data can be mapped precisely. This is due to the latency time and the low revised sampling rate of a maximum of 10 Hz of the AM machine data, as the repetition rate of the scanning heads is up to several tens of kHz in comparison. The values of the machine data points can only be registered to multiple scan vectors and their timestamps of a cross-section of one part within one layer. All scan vectors within a time range are assigned to the sensor value valid for this range. A method for retrieving the pre-computed timestamps of the fabrication and the applied parameters in a runtime-independent format via a Software Developer Kit (SDK) is presented in the next section.

2.2 Extraction of scan vector data via SDK

Since the smallest entity for data registration in the proposed ETL pipeline is a scan vector, it is essential to extract this data from each build cycle.

Nevertheless, it is neither available via the OPC UA nor the REST API. The OPC UA protocol is not designed to transfer the data of the scan vectors in parallel with the fabrication in terms of signal cycle times and signals per second. Additionally, as discussed earlier in this section, there is no interface implemented on the applied PBF-LB/M machine, where for instance, a custom REST API could retrieve this data from and provide it asynchronously to the fabrication. However, the machine is executing the scan vectors during the build cycle, so the information must be available. To access them, the build cycle file, for EOS in a.openjz format, gets parsed prior to fabrication. It contains the geometries of the parts, the parameter set and the instructions on how to create a format that can be executed by the PBF-LB/M machine. The task of slicing and mapping the parameters on each scan vector, depending on its section within the part, is performed via an SDK [16].

The PBF-LB/M machine that is applied operates a priori. As a result, the entire build cycle is pre-computed in advance. In this work, it is processed layer by layer and converted into JavaScript Object Notation (JSON) format by an external Python script. Thereby the start and end points of each vector are accessible. This information corresponds to what will be executed on the machine, along with the laser parameters applied during fabrication, assuming no process

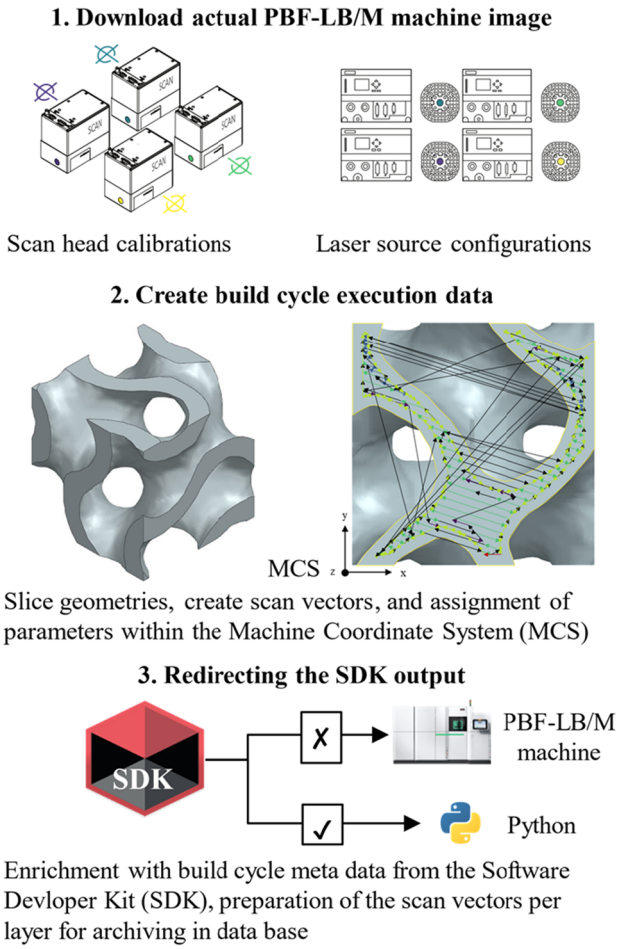

Fig. 4 Scan vector extraction of a build cycle

deviations. To ensure accuracy, a current image of the AM machine, which includes the calibration of the scan heads and laser sources, is retrieved before compiling the instructions for fabrication. A coarse program workflow is given in Fig. 4, an exemplary output is given in Sect. 4.4.

3 Data pipeline concept

The key requirement for the ETL pipeline is to capture the process signals in situ and register them on the process entities, such as the AM machine, build cycle, part, layer, and scan vector. To achieve this, the machine data from the fabrication are accessed via the OPC UA server of the AM machine and the corresponding scan vectors data is extracted via an SDK. Both data sources must be integrated into a single ETL pipeline. Since the OPC UA server represents a dynamic data source [17], the concept of streaming the data in five stages is applied, see Fig. 5. The first stage,

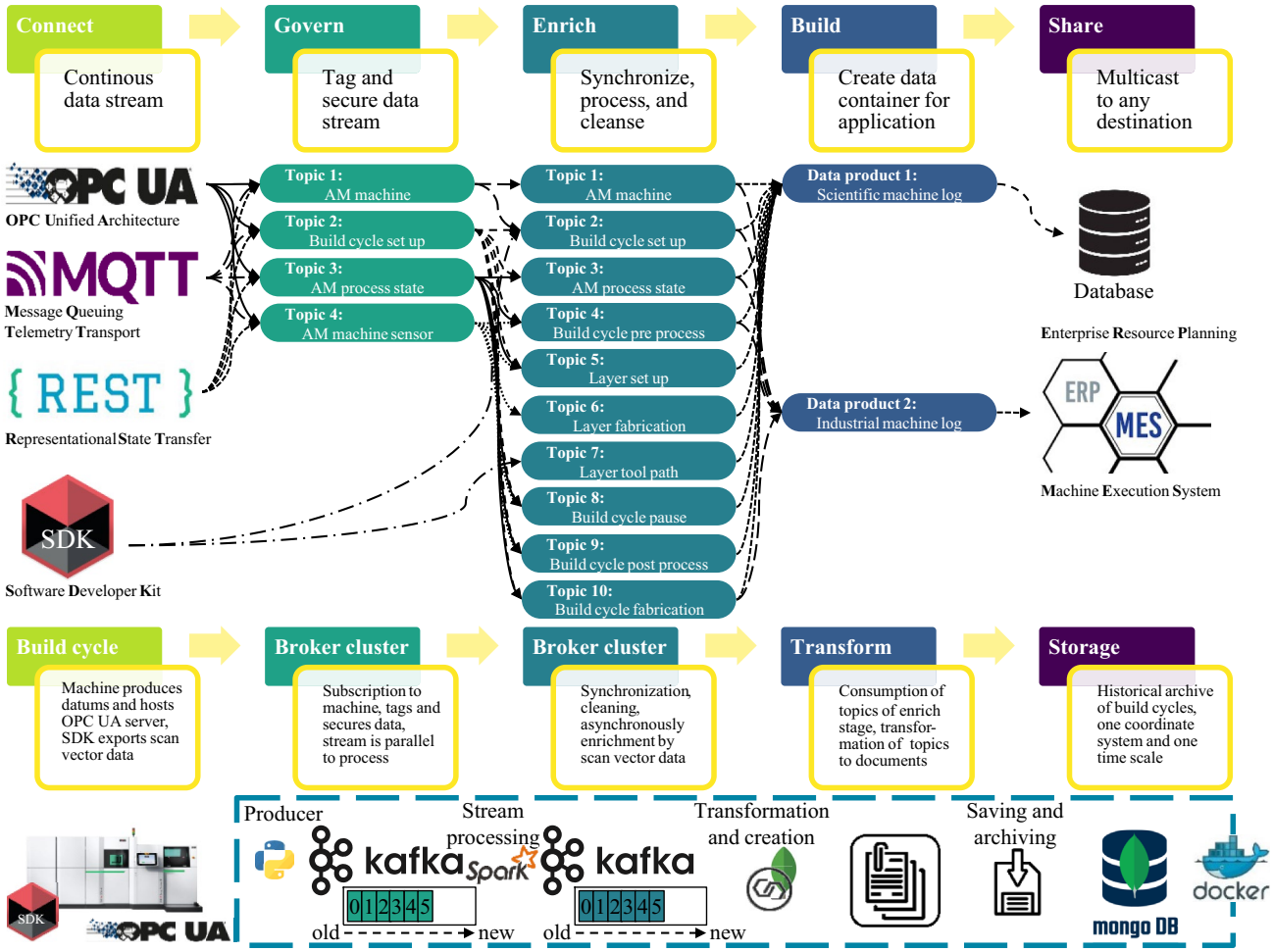

Fig. 5 Stateful ETL pipeline concept and setup with images from [16]

also called connect stage, involves authenticating the client and establishing a robust subscription to the machine datums. In the second stage, also known as the govern stage, the data stream is organized into topics via a distributed event streaming platform. Here the loosely published data points, provided only with an identifier and timestamp, are aggregated. This process tends to have a maximum offset in the range of seconds, as shown in Fig. 3. The distributed event streaming platform performs the initial caching via the brokers of its cluster and maps the data points based on their identifier to the topics "AM machine", "Build cycle setup", "AM machine sensor", and "AM process state", see Fig. 5. Utilizing this concept, a cardinality is introduced. The machine datums of topic 1 and topic 2, shown in Fig. 5, remain static once a build cycle is initialized. Serial numbers of soft- and hardware components are then constants since a change of this data is unlikely during fabrication. This also reduces the data traffic. However, the topics of AM machine sensor and AM process state present a different scenario. The machine data associated with these topics are linked to the fabrication and fluctuate within certain limits randomly. This variability is inherent to the fabrication and is also expected under normal operating conditions. Therefore, the data points need to get further processed and aligned to one common time scale with fixed increments. Otherwise, it is not possible to register and link data measured with different revised sampling rates on the process entities. This is particularly true when the threshold for a value has not been exceeded for a longer duration and no new value has been published. To address this issue, a common time scale is established from the start. The size of increments is governed by the highest revised sampling rate of all machine datums. All machine datums with a lower sampling rate, or those that did not change and are therefore not republished, are maintained as valid by the concept of resampling with forward fill. This ensures that a request by a user at

the database will access all relevant information without the need to manually query the otherwise missing values and apply domain knowledge to do so.

This stream processing takes place in a distributed computing framework between the govern and the enrich stage from Fig. 5. It assures that the topics in the enrich stage are already consisting of agglomerated data points for the respective entity. The distributed event streaming platform, again hosting these topics via the brokers, its cluster receives the data asynchronously to the process itself due to the computation time of the resampling. However, this modular concept provides near real-time data at the governing stage and processed data sets at the enrichment stage, where external sources like the output of an SDK can be coupled in. The enriched data is then organized into a data product, in this case a sink, which serves as an archive in the proposed ETL pipeline. This data product allows the structured storage of a build cycle. It is structured in a way that queries for various entities, such as a specific layer and the valid conditions during its fabrication, like the torque curve for recoating, can be performed efficiently. For this functionality and to support the claim of understanding with the lowest possible amount of domain knowledge, a No Standard Query Language (NoSQL) database schema is engineered, as shown in Fig. 6. The database where the schema is implemented is document oriented. This approach is particularly suitable for PBF-LB/M and its high degree of freedom of process characteristics, as it allows data to be organized without the need for bridging tables [18, 19]. In a relational SQL database, these tables, which serve solely as links and contain no additional information, are a requirement. In the NoSQL database schema of Fig. 6, instead, each table represents a collection of documents, as indicated by bold entries. Each document stores the values of the assigned OPC UA nodes, which are processed and forwarded trough the topics of the enrichment stage. The cardinality is established through the time stamp aligned with the timescale of the build cycle, the process states of the AM machine, and the layer identifier. This approach offers the adaptability needed for altering the data schema or incorporating new data sources. For instance, it simplifies the extension by a new data source by allowing the creation of a new collection, thus avoiding the complexities associated with a rigid SQL database. To facilitate the sharing aspect of the data product at the last stage of Fig. 5, the NoSQL schema used to register the stream during the build stage is designed with established semantics and, wherever possible, ontologies are considered.

Initially, all terms from the ISO/TC 261—Additive Manufacturing standards are collected by web scraping the official International Standard Organization (ISO) Online Browsing Platform (OBP), where term definitions can be viewed in the preview of each standard. This list is then compared with all terms from the National Institute of Standards and Technology (NIST) AMS 500–1 [20] in addition to the F3490 -21 [21], also including their ontology. The data schema engineered in this work contributes to filling the gap of a standardized machine log file, identified in the standardization roadmap [22].

By maintaining the same ontology for already described relations, such as the AM machine and build chamber, along with the semantics, interoperability with the stated Common Data Exchange Formats (CDEF) [20] is achieved. The remaining open term definitions, especially in the field of AM process states, are taken from the OPC UA release candidate 40001–3 [23]. Utilizing established terminologies and ontologies like for CDEF, the schema is user-friendly and easily understandable. This approach also provides a solid foundation for further inquiries, enhancing scientific integrity of the schema.

4 Data pipeline setup

The ETL pipeline is configured as a standalone Docker application. This design ensures independence from the computational system on which it is executed [19]. Supplementary, it enables sharing of the host system resources, making it more efficient than a Virtual Machine (VM). The implementation of the ETL pipeline, depicted at the bottom of Fig. 5, is composed out of microservices. Each microservice operates within its own container, orchestrated by Docker Compose, which establishes connections between the containers by opening the necessary ports. This approach facilitates inter-container communication and enhances the overall efficiency of the application. At the beginning of the pipeline, a Python producer retrieves data from the PBF-LB/M machine via the OPC UA server and transmits it to the brokers of the Apache Kafka cluster, see Fig. 5. The records of these topics of the govern stage are subsequently relayed to PySpark, where they are further processed, as explained in Sect. 3. Thus, the structured data gets then extended by the output from the SDK and is cached again at the Kafka cluster in the topics of the enrich stage. Ultimately, a connector to MongoDB is employed to perform type casting and store the process signals as a machine log. The microservices and especially the triggers of the ETL, which are controlling the logic of tasks such as the assignment of nodes based on their

values to different topics, are discussed in detail in the following subsections.

4.1 ETL pipeline triggers

To initiate the pipeline, the Python producer subscribes to the node of the OPC UA server that indicates whether the PBF-LB/M machine is operational. When the state changes to true, the pipeline subscribes to all relevant nodes of the OPC UA server. For further processing between the govern and the enrich stage, as shown in Fig. 5, additional logic is required. Wherever possible, in accordance with the namespace of OPC UA 40001–3 [23], basic process states and events are declared to describe a unique subprocess during a build cycle. This enables the mapping of process signals to process entities. Therefore, the six machine states "AM machine idle", "Build cycle pre-process", "Build cycle processing", "Build cycle error", "Build cycle pause", and "Build cycle post process" are defined for PBF-LB/M. These states, in conjunction with the identifier of each layer and the declaration of the result of each build cycle, are establishing a set of variables for uniquely defining each topic in the ETL pipeline. As a result, the pipeline must exhibit conditional behavior, often referred to as stateful. The optimal implementation of each condition is achieved by minimizing the logical terms of each process state. This is accomplished utilizing the declaration of each process state, based on domain knowledge, as a starting point. The Espresso Algorithm, a heuristic logic minimizer [24], is then used to carry out this optimization via the implementation of the Python library pyeda. The pipeline, depicted in Fig. 5, is triggered by this logic and autonomously performs the agglomeration and structuring into the topics. This process is based on the values of the OPC UA nodes, or more specifically, the edges of their resulting signals. It is tested and validated against a VM simulating the EOS M 300–4, which is hosted on Microsoft Azure and creates mock data.

4.2 Producer

The OPC UA client is implemented using the Python library opcua-asyncio, as detailed in Sect. 2.1. The subscription of the client is configured through a Python script that queries the available nodes of the OPC UA server. It registers the relevant nodes to the terms from the database schema, of Fig. 6. These nodes are then incorporated into the pipeline through JSON configuration files for the stream set-up.

4.3 Broker cluster govern stage

The Kafka cluster hosts the four topics from Fig. 5. It is responsible for receiving incoming messages from the Python producer, storing, and serving them to the streaming processor PySpark synchronously.

Messages are stored in a key–value format, where the key is the node id of the OPC UA schema, and the value is the measurand of the OPC UA node. These records are stored along with a timestamp in topics. The topics are distributed and replicated across the brokers of the Kafka cluster to ensure fault tolerance and enable load balancing. Within a partition, records are stored in the order they are written. The Kafka cluster of the test bed uses 1 partition per topic with a replication factor of three, striking a good balance between redundancy, storage resources and future scalability for a fleet of machines. The maximum retention is set to two weeks. This implies that in the event of a sink failure, the data can be recovered within this timeframe.

4.4 Streaming processor

The primary logic of the pipeline, in conjunction with the assurance of data integrity through resampling as outlined in Sect. 3, is executed utilizing micro batching, implemented via PySpark. Depending on the state of the build cycle and the AM machine, which is represented by specific combinations of the values of the OPC UA nodes, the stream is asynchronously divided into the topics of the enrich stage. Records within these topics are structured in a way that facilitates their registration on process entities, such as a layer. In PySpark, the enrichment of information through scan vectors is conducted, as the output of the SDK is coupled into the pipeline. This integration imports the corresponding scan vectors for each layer of the build cycle, as well as the settings for each part. An extract of the SDK output, which is prepared to align with the data schema, is presented below for one part, its initial layer, and the first scan vector of this layer.

```
{"1": {
      "beam_compensation": "0",
      "hatch_origin": "MCS",
      "id": "2023_10_30_Gyroid",
      "process_parameter_set":
"AlSi10Mg_060_CoreM304_102_Default_DirectPart",
      "scaling_x": "0",
      "scaling_y": "0",
      "translation_x": "0",
      "translation_y": "0"}
}
{"layer_1": {
      "delta_z": 60,
      "exposed_part": ["2023_10_30_Gyroid"],
      "sdk_id": 2023-29-14 20:15:15,
      "position_z": 60,
      "vector_1": {
        "end":   {"x_mm": 152.343,
                  "y_mm": 148.447},

        "exposure_unit": 0,
        "inter_vector_time": 0.0,
        "laser_power": 315.25,
        "part_id": "2023_10_30_Gyroid",
        "scan_speed": 1210.0,
        "start":  {"x_mm": 152.390,
                   "y_mm": 148.544},
        "timestamp": 1640.0,
        "type": "INFILL"},
        }
}
```


4.5 Broker cluster enrich stage and sink

The topics of the enrich stage of the Kafka cluster are serving as modular instances, extending the pipeline for potential integration of external data sources like an MPM, given that the connection to the machine log is already established. Additionally, they are natively feeding data to the corresponding collection of the NoSQL database MongoDB via a Kafka connector. This setup provides high flexibility, enabling adjustments in the scope of the extracted and permanently stored data, to realize different data products, as shown in Fig. 5. The schema of the data product for this work is designed using Hackolade, a NoSQL schema modeler. This tool facilitates a swift iterative design process through forward-engineering. The database schema is illustrated in a compressed form in Fig. 6, excluding the nested documents within collections. This schema circumvents the 16 MB per document limitation of MongoDB and enables selective data retrieval during querying the database. In addition, it can accommodate data with high variance, such as the number of components and layers.

5 Conclusion and outlook

The proposed ETL pipeline establishes a seamless connection from the SCADA to the MES level. The scalable and extendable data pipeline, made robust through Docker, ensures consistent operation across all environments. This pipeline can accommodate the three main IoT communication architectures for data acquisition. It requires adjustments only to the Python producer to establish the connection and retrieve the data, as explained in Sect. 2.1. This flexibility allows it to address various network technology limitations, such as available bandwidth or stability.

The ETL pipeline empowers users to customize data acquisition based on their application needs, by configuring their own data product. For example, a real-time dashboard can be mounted on the govern stage due to its low latency. In the current test bed, the data provides valuable and easily interpretable insights into machine health and production operability, detailed at the scan vector level. Additionally, Key Performance Indicators (KPIs), such as the Overall Equipment Effectiveness (OEE), can be derived. The provision of data in a runtime-independent format supports the FAIR principles (Findable, Accessible, Interoperable, and Re-usable) for PBF-LB/M. Therefore, the domain knowledge required to evaluate the data is reduced. No specific software is mandatory to access the fabrication data or retrieve the image of the AM machine of the fabrication. The only flaw in this work is that currently the timestamps are pre-computed for each layer. This procedure is assuming an ideal process with no additional delays, like the cleanse of the filtration system within a build cycle.

It can be resolved by retrieving the actual timestamp of each scan vector through access to the traffic of the SL2-100 protocol, see Fig. 1. Therefore, discrepancies between the instructions and the fabrication itself, as also reported in the literature, can be qualified [25]. By substituting the now pre-computed timestamps against the real timestamps, the data quality can be further increased on the level of scan vectors. This extension, as well as the incorporation of MPM data, will be carried out in future.

Appendix

NoSQL schema of the PBF-LB/M machine log

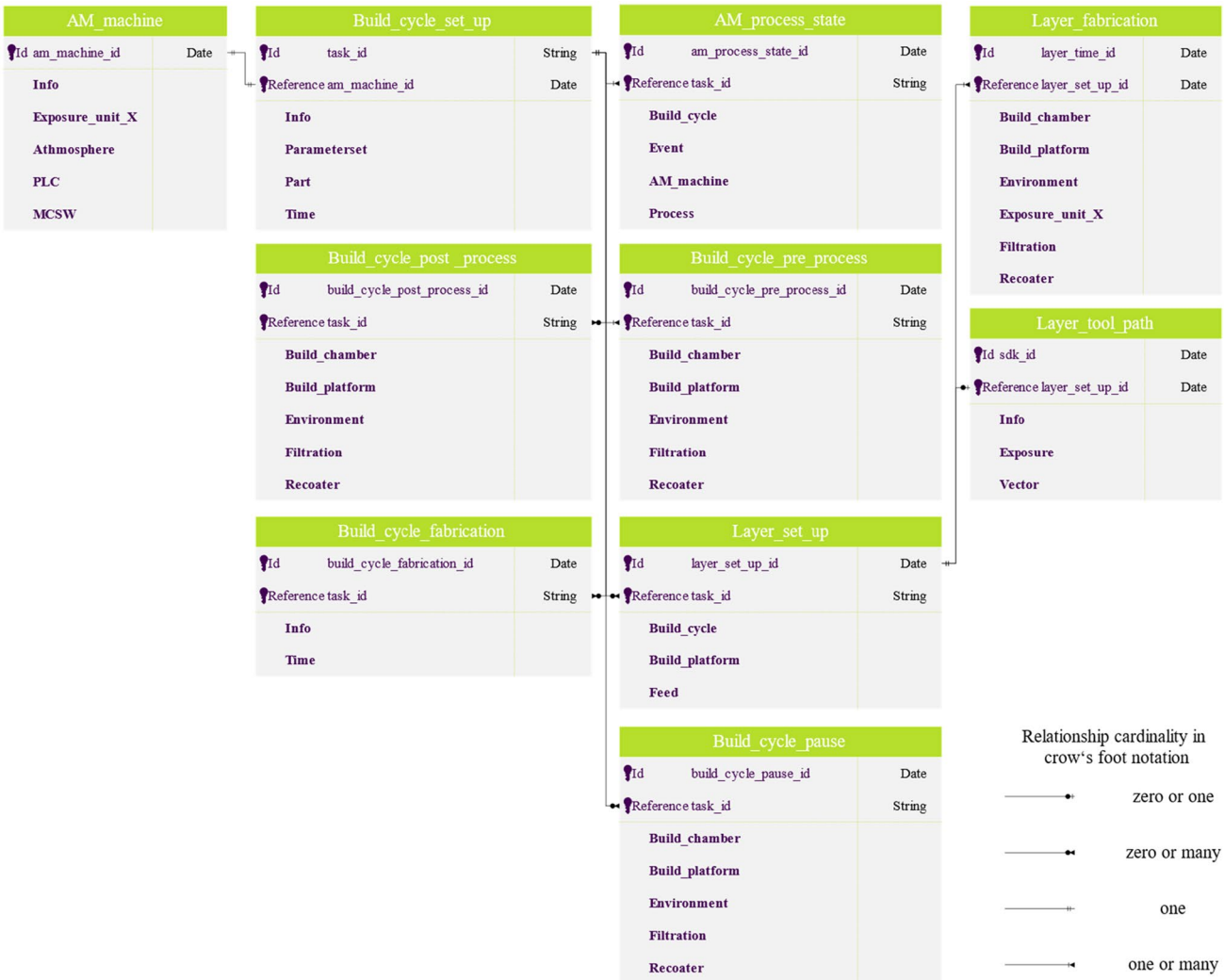

Fig. 6 NoSQL data base schema and cardinality of the MongoDB instance down to the level of first order documents

Acknowledgements Gratitude is extended to Pascal Desmarets for providing guidance for the use of Hackolade, which expedited the data schema design process for MongoDB and facilitated an iterative and collaborative approach through forward-engineering. The authors also acknowledge the contributions of Michael Scharf from EOS, who generously shared his expertise on the SDK and provided support for validating the ETL pipeline via the VM of the EOS M 300-4, as part of the EOS Developer Network (EDN).

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability Selected code components and data records will be made available on request.

Declarations

Conflicts of interest The authors declare that there is no conflict of interest for the present work.

References

1. Wohlers TT, Associates W, Campbell I, Diegel O, Huff R, Kowen J, Mostow N, Staff WA. (2021). Wohlers Report 2021: 3D printing and additive manufacturing global state of the industry. Wohlers Associates, Incorporated
2. Bartsch K, Pettke A, Hübert A, Lakämper J, Lange F (2021) On the digital twin application and the role of artificial intelligence in additive manufacturing: a systematic review. J Phys Mater 4:032005
3. Yang H, Rao P, Simpson T, Lu Y, Witherell P, Nassar AR, Reutzel E, Kumara S (2021) Six-sigma quality management of additive manufacturing. Proc IEEE 109:347–376
4. Zhang Y, Safdar M, Xie J, Li J, Sage M, Zhao YF (2022) A systematic review on data of additive manufacturing for machine learning applications: the data quality, type, preprocessing, and management. J Intell Manuf. https://doi.org/10.1007/s10845-022-02017-9
5. Zhang L, Chen X, Zhou W, Cheng T, Chen L, Guo Z, Han B, Lu L (2020) Digital twins for additive manufacturing: a state-of-the-art review. Appl Sci 10:8350
6. Kuhn T, Antonino PO, Schnicke F. (2020) Industrie 4.0 virtual automation bus architecture. In: Software Architecture: 14th European Conference, ECSA 2020 Tracks and Workshops, L'Aquila, Italy, September 14–18, 2020, Proceedings 14, pp. 477–489. Springer,
7. Höfflin D, Sauer C, Schiffler A, Hartmann J (2022) Process monitoring using synchronized path infrared thermography in PBF-LB/M. Sensors 22:5943
8. Standardization IOf. (2022) DIN CEN ISO ASTM TS 52930 April 2022 Additive fertigung grundlagen der qualifizierung installation funktion und leistung (IQ OQ PQ) von PBF-LB Anlagen. vol. DIN CEN ISO ASTM TS 52930.
9. eV DIfN. (2016) Referenzarchitekturmodell industrie 4.0 (RAMI4.0). DIN SPEC 91345,
10. Profanter S, Tekat A, Dorofeev K, Rickert M, Knoll A. (2019) OPC UA versus ROS, DDS, and MQTT: performance evaluation of industry 4.0 protocols. In: 2019 IEEE International Conference on Industrial Technology (ICIT), IEEE.
11. Standardization IOf. (2021) DIN EN IEC 62541–5 August 2021 OPC unified architecture teil 5 informationsmodell. DIN EN IEC 62541–5.
12. Morato A, Vitturi S, Tramarin F, Cenedese A. (2020) Assessment of different OPC UA industrial IoT solutions for distributed measurement applications. In: 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC). IEEE,
13. Anlagenbauer VDM-u. (2023) OPC UA for Additive Manufacturing. VDMA 40540.
14. Gamper S, Poudel BK, Schriegel S, Pethig F, Jasperneite J (2020) Untersuchung der Netzlastrobustheit von OPC UA-Standard, Profile, Geräte und Testmethoden. Kommunikation und Bildverarbeitung in der automation: Ausgewählte Beiträge der Jahreskolloquien KOMMA und BVAu 2018. Springer, Berlin Heidelberg
15. Garg H, Dave M. (2019) Securing IoT devices and securely connecting the dots using REST API and middleware. In: 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU). IEEE.
16. Poka K, Merz B, Epperlein M, Hilgenberg K. (2023) Integration of the whole digital chain in a unique file for PBF-LB/M: practical implementation within a digital thread and its advantages. In: International Conference on Additive Manufacturing in Products and Applications. Springer.
17. Burger A, Koziolek H, Rückert J, Platenius-Mohr M, Stomberg G. (2019) Bottleneck identification and performance modeling of OPC UA communication models. In: Proceedings of the 2019 ACM/SPEC International Conference on Performance Engineering.
18. Aggour KS, Kumar VS, Cuddihy P, Williams JW, Gupta V, Dial L, Hanlon T, Gambone J, Vinciquerra J. (2019) Federated multimodal big data storage & analytics platform for additive manufacturing. In: 2019 IEEE international conference on big data (big data). IEEE.
19. Michalkowski C, Janhsen J, Springer P (2023) Concept for a generic modular software architecture for the integration of quality relevant data and sample implementation for a laser sintering system. Prog Addit Manuf 8:67–73
20. Li S, Lu Y, Aggour K, Coutts P, Harris B, Kitt A, Lupulescu A, Mohr L, Vasquez M (2023) Enabling FAIR data in additive manufacturing to accelerate industrialization. Boulder, Colorado, National Institute of Standards and Technology
21. Materials ASfTa. (2022) ASTM F3490–21 Standard practice for additive manufacturing—general principles—overview of data pedigree. ASTM F3490–21,
22. (AMSC), AMaAAMSC. (2023) Standardization roadmap for additive manufacturing, Version 3.0.
23. Anlagenbauer VDM.-u. (2023) OPC UA Job management. VDMA 40001–3

24. Rudell RL, Sangiovanni-Vincentelli A (1987) Multiple-valued minimization for PLA optimization. IEEE Trans Comput Aided Des Integr Circuits Syst 6:727–750
25. Duong E, Masseling L, Knaak C, Dionne P, Megahed M (2022) Scan path resolved thermal modelling of LPBF. Addit Manuf Lett 3:100047