**Solid Freeform Fabrication 2021: Proceedings of the 32nd Annual International Solid Freeform Fabrication Symposium – An Additive Manufacturing Conference**
**Reviewed Paper**

# A Data Integration Framework for Additive Manufacturing Big Data Management

Milica Perišić[1], Dimitrije Milenković[2], Yan Lu[3], Albert Jones[3], Nenad Ivezić[3], Boonserm Kulvatunyou[3]

[1]Faculty of Organizational Sciences, University of Belgrade, Serbia
[2]TX Services, Serbia
[3]Systems Integration Division, National Institute of Standards and Technology, Gaithersburg, MD

## Abstract
Large amounts of data are generated throughout the entire, AM, part-development lifecycle. Data are generated by various functions within process monitoring, material characterization, equipment status, and part qualification. Hence, data integration and management are critical in streamlining, accelerating, certifying, and deploying these functions. However, achieving that integration and management has several challenges because AM data embodies the four characteristics of Big Data - volume, velocity, variety, and veracity. This paper proposes an AM framework as a foundation for addressing those challenges. In the framework, AM data are streamed, curated, and configured automatically for real-time analysis and batch processing, which increases the effectiveness of archiving and querying that data. The framework also includes a description of the associated AM metadata, which links the various data types and improves browsing, discovering, and analyzing that data. Finally, the framework can be used to derive requirements for standards that enable data sharing.

## 1. Introduction
Large amounts of data are generated through the entire AM development lifecycle. Data is generated and collected for material characterization, process monitoring, part qualification, etc. Hence, integrating and managing this data is critical in streamlining, accelerating, certifying, and deploying those functions. However, successfully integrating and managing this data introduces several new challenges because AM data embodies the four characteristics of Big Data - volume, velocity, variety, and veracity. Moreover, addressing these challenges is critical for advancing the capabilities and use of AM production technologies.

Today, AM technologies are mainly used for rapid prototyping; but they are slowly emerging as a commonly used production technology in several industry sectors [1]. AM data integration is one of the top roadblocks to completely automated production management. A major reason for this roadblock is the lack of established methods and standards that allow a quick, "plug-N-play" type of integration of AM data sources with various manufacturing and enterprise applications. For example, the large amounts of high-speed, in-process, monitoring data such as melt pool images cannot be acquired, and automatically processed for real-time, or near-real-time control.

A "divide and conquer" approach is proposed in this paper to create a data integration framework that addresses these issues. The framework has seven functional steps, which are discussed in the next section, including Defining Dataset/Data Source, Collecting Data, Queueing Data, Archiving Data, Downgrading Data Amount, Building Decision Models, and Using Decision Models. In addition, the paper offers a guide for using the framework with the current industry standards.

## 2. A Big Data Integration Framework

Figure 1 presents a framework that can facilitate the creation of a big data integration system that would enable real-time monitoring and control and long-term data archiving for offline analyses. The framework consists of seven steps. Step one provides a clear definition of data that comes to the system and ensures that future misunderstandings between the data provider and the system maintainer would be avoided. Step two collects or receives the data from the data provider and enters it into the system to be further processed. Step three prevents the system from being overwhelmed by queueing raw data until it can be processed and stored in the system. Step four decides how the data is stored, which is the key to having a stable and tenable system that makes data accessible both using queries and bulk downloads. Step five achieves a sustainable system by reducing and managing the data size and saving only the data necessary for future use cases. Step six creates new value by building Decision-Making algorithms and models that use the stored data as inputs. Step seven creates programs that automatically allow use of the system through alarms, monitoring reports, and automatic actions. Each step is discussed separately with the aim to emphasize the new standards that are needed.

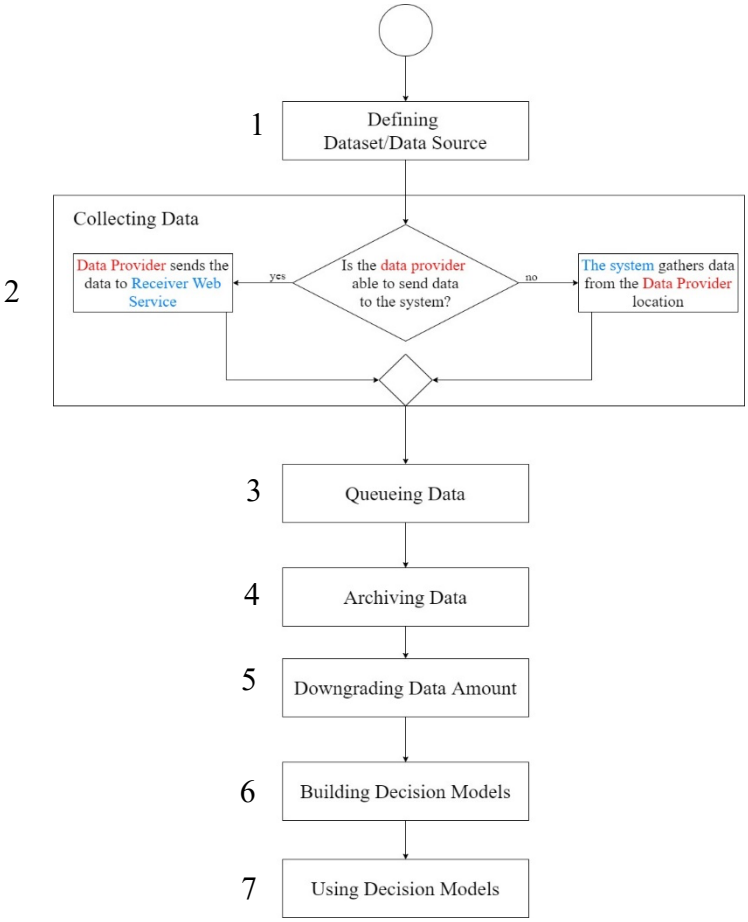


Figure 1. A big data integration framework

### 2.1 Defining a Data Source

A clear definition of a data source and its data is critically important both for the Data Provider and for the System Maintainer because it ensures that both parties have the same understanding of that data. Such a definition represents a joint agreement between the Data Provider and the System

Maintainer. Moreover, if this definition is not standardized and defined, it can cause misunderstandings throughout the whole life cycle. Three important descriptions are needed to create a standardized definition.

**2.1.1 Data-Source description:** Data-source description provides the information about the device that generates the data as well as the measurement associated with the data. "Data Source Name" specifies the name of the data source. "Data Source Type" is necessary to indicate its physical characteristics. For example, the "Data Source Type" can be "*Coaxial Camera-based MPM*", or "*Layerwise Overview Imager*" etc. "Device Type", "Device ID", "Device Manufacturer", "Device Model" and "Device configuration" captured the metadata of the measurement devices or actuation devices. In the case of the co-axial-camera-based, melt-pool, monitoring system, the device type could be "CMOS Camera". The metadata definition for data sources can be leveraged on Dublin Core Metadata Initiative schema[1].

**2.1.2 Data description:** Another block of information is about the data itself. Typical attributes include "Data Category", e.g., Sample, Event, or Condition based on MTConnect[2]. Data Type can be Value, TIME SERIES, and DATA SET as defined in MTConnect. Additional data types include images, videos, and 3D models, etc. Coordinate System is also a data field that could be very important to interpret position and size of measurement data.

**2.1.3 Load description:** To integrate data sources, it is important to know whether the data are from streaming or batch upload as well as additional metadata about the load, e.g., size and frequency. For example, consider an image data type. Will an image file and its metadata be sent as one object or two? What's the expected size? Approximately, how many data instances will be sent per day or per batch? What's the communication protocol? In the case of streaming data sources, some additional information is needed, such as sample interval and streaming triggering mechanism.

Integration of streamed data is more challenging compared to the batch upload, so that will be the focus of the paper. Nevertheless, the framework can be applied to batch uploads as well.

### 2.2 Collecting Data

After defining the data, it is necessary to decide how the data will be collected from the data source. In most processes, including Additive Manufacturing processes, in-process data is created by following some of these three policies: sample, on event, and on condition. This means that data will be created during a defined sample interval, when a defined event occurs, or when a defined condition is satisfied, respectively. Independently of how the data is created, there are two main approaches when it comes to the data collection – push or pull. These approaches can be applied for both streaming and batch data processing.

In the first case, the Receiver Web Server, shown in Figure 1, is a data gateway. At this step, it is important to define a standard for exchanging the data. The Receiver Web Server is an application written in Java or some other programming language that receives the data through HTTP/HTTPS protocols and forwards it to the queue. Queue and its purpose are discussed in the next step of the framework.

In the second case, when the data provider, the device, cannot push the data to the system, the approach consists of developing an application that will pull data from the data provider's specific

[1] https://dublincore.org/schemas/
[2] https://www.mtconnect.org/standard20181

location and forward it to the queue. The best practice is to define a standard location for both, the data, such as images created during an AM process, and its metadata. The application can use some of the standard protocols to pull data such as SSH and FTP. The application can be created using Shell or any other scripting language and can be scheduled using CRON[3]. Also, for this case some recent technologies could be used, such as Apache NiFi[4].

### 2.3 Queueing Data

Processing a data instance requires system resources. To prevent the system overload, a message queue can be helpful. In that way, data will be temporarily stored in the queue until it is processed and stored in the system. Multiple queues may also be used. For example, the first queue stores raw, unprocessed data waiting to be processed. The second temporarily holds processed data waiting to be stored in the system. Common message queue technologies used for these purposes are Apache Kafka[5] and IBM MQ[6].

### 2.4 Archiving Data

Deciding on which persistent-storage technology to use is key to having a stable and tenable system. Metadata and image data should be defined and stored separately. It is common to use file systems as image storage, but metadata can be stored in a database that can be searched and queried quickly and easily. Following that approach, metadata will have one additional field - image location from where the image can be read.

**2.4.1 Metadata Storage:** Some of the options for storing metadata are a relational database, a file system (e.g., Hadoop Distributed File System - HDFS[4]), or a document database (e.g., MongoDB[7]). Relational database management systems (RDBMS) are the industry standard. The main advantage of RDBMS is that most engineers know this technology and there are integrations with many external tools. Considering that metadata will not change after insertion, document databases can be used as well. However, the file system is appropriate only if data queries are based on few fields with a limited number of values. When this is the case, these fields and values can be used to partition the metadata within a file system. This approach allows faster metadata access.

#### 2.4.2 Image Storage

**2.4.2.1 File System Storage:** One way to store big data, including images, is to use a file system. Currently, a commonly used one for big data is Hadoop Distributed File System (HDFS[8]). HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware. In recent years, many other tools and solutions were developed to support Hadoop core components. All of them together make data analysis, storage, and maintenance of data easier.

**2.4.2.2 Cloud Storage:** Another convenient way to handle images, and other binary objects, is to shift responsibility to cloud-based storage centers. Storage-as-a-service provides features to easily scale computational resources and provide access permissions. It also offers a user interface, command-line tools, and an API for several programming languages.

---

[3] https://en.wikipedia.org/wiki/Cron

[4] https://nifi.apache.org/

[5] https://kafka.apache.org/

[6] https://www.ibm.com/products/mq

[7] The most popular database for modern apps | MongoDB

[8] https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html

**2.4.2.3 Database Storage:** Database storage is another possibility. Here, the data, including images, with the associated metadata can be stored in one schema, which is the biggest advantage of this approach. Nevertheless, this approach is more appropriate for a small number of images and for applications that need to search data in real-time.

The decision of what kind of storage to use depends on the target use case. If there are no obstacles in uploading images to cloud-managed storage and it's possible to pay for the external software the choice should be cloud-managed storage. It is easier to maintain and there is no need to worry about space limitations. On the other hand, if data is private and should be kept in local systems then paid cloud storage may not be a good option. A file system storage may be a better fit in this case. However, it is one that requires more resources and skills to set up and later administrate. On the other hands, a file system leaves more freedom to customize and adjust the solution to specific target use cases.

**2.4.3 Partitioning:** Regardless of the choice of local or cloud-based file storage, special attention should be paid to partitioning. Partitioning data is crucial because it can drastically affect data processing. By partitioning data, the amount of data scanned by each query is smaller, thus improving performance and reducing cost. A common practice is to partition the data based on time, often leading to a multi-level, partitioning scheme; but many other attributes may be used.

### 2.5 Downgrading data amount

To achieve sustainability, the system needs to be capable of managing different data sizes and save only the data necessary for future use cases. Several policies need to be defined: deleting the old data, aggregating the data, removing duplicates, and reducing data quality if needed.

### 2.6. Building Decision Models

To gain value from the data, many possibilities exist. One is to create an AI-based, decision-making model and another is to build a rule-based, expert system. AI-based models include predictive and clustering models. Predictive models need a solid history of data labeled with the information that needs to be predicted in the future. Clustering models create a group of similar data items that help experts use to support decision making. Rule-based expert systems are built using if-then rules, which are defined on the top of the fields of the data instances. Rules are defined by the experts in the specified area. Common practice is to combine both AI-based and rule-based approaches to build reliable, decision-making systems.

### 2.7. Using Decision Models in Practice

There are several options on how decision-making models can be used in practice. In Additive Manufacturing, for example, if a model can predict a critical event, an alert can be sent to staff in charge of handling that issue. If the model reports an in-process, monitoring anomaly, it can be used to change process parameters or stop the build.

## 3. In-process Data Integration for Additive Manufacturing

A variety of commercially available sensor technologies are now used in monitoring and controlling AM laser-based powder bed fusion (LBPF) processes. These technologies can be classified into two types: local sensors and global sensors. Local sensors have smaller fields of observation and acquire data with finer spatial resolutions and/or higher sampling rates. Global sensors often monitor the whole build surface and acquire data at lower resolutions. The two types complement each other and are used together for monitoring AM processes and predicting build quality [2].

Camera-based, melt-pool monitoring (MPM) involves local sensing which generates high-resolution, melt-pool images at a high sampling rate. MPM systems are critical for deep process understanding, in-process defect detection, and real-time control. Hence, they are one of the most attractive solutions for LPBF in-process monitoring [3]. However, MPM data streaming and integration for both real-time processing and offline analysis is very challenging for two reasons: the high velocity of the data generations and the big volume for archiving. In the following sections, the proposed data-integration framework is applied to MPM to illustrate and validate its benefits.

### 3.1 Applying the Framework to AM

**3.1.1 Requirements:** To implement the framework for this use case, the requirements are first collected. The main requirements include knowledge of the input data, the capabilities of the system, the availability of storage, and standards for data flows. According to the framework, we obtained data source definition. Example input data is given in the table below.

Table 1 Data Source Definition for MPM

| Metadata Field | Data Definition | Value |
|---|---|---|
| Data Source Name | Data-Source description | AMMTCoAxialMPM_Mikrotron-EOSens-3CL |
| Data Source Type | | Coaxial Camera-based MPM |
| Data Source Description | | NIST AM Metrology Testbed[9] |
| Device Identifier | | Mikrotron-EOSens-3CL |
| Device Description | | Mikrotron, EoSens ® 3CL, "http://www.mikrotron.ir/Datasheet/mikrotron_eosens_3cl_dsh_03.pdf" |
| Device Type | | CMOS Camera |
| Device Parameters | | Image Size, Pixel resolution - 15KB, 120x120 |
| Data Category | Data description | Sample |
| Data Type | | Melt Pool Image |
| Describe By | | BMP/PNG |
| Trigger Method | Load Description<br><br>Load Description | PUSH |
| Time | | Start Time/Stop Time |
| Protocol | | Camera Link |
| Sample Interval | | 50 microseconds |
| Measurement Setting | | Magnification X1 |
| Calibration Information | | Fully Corrected |

[9] https://www.nist.gov/el/ammt-temps

The next requirement is knowledge about the sensor data transfer capability- is it possible to send an image or not. Based on that, an application of the framework will proceed differently.
The last prerequisite is to provide storage where the data will be stored. Since the framework proposes separate storage for metadata and images, it is necessary to provide both.

**3.1.2 The Architecture:** Figure 2 shows the architecture we propose for AM MPM data integration. The following sections describe how we use the framework with this architecture.

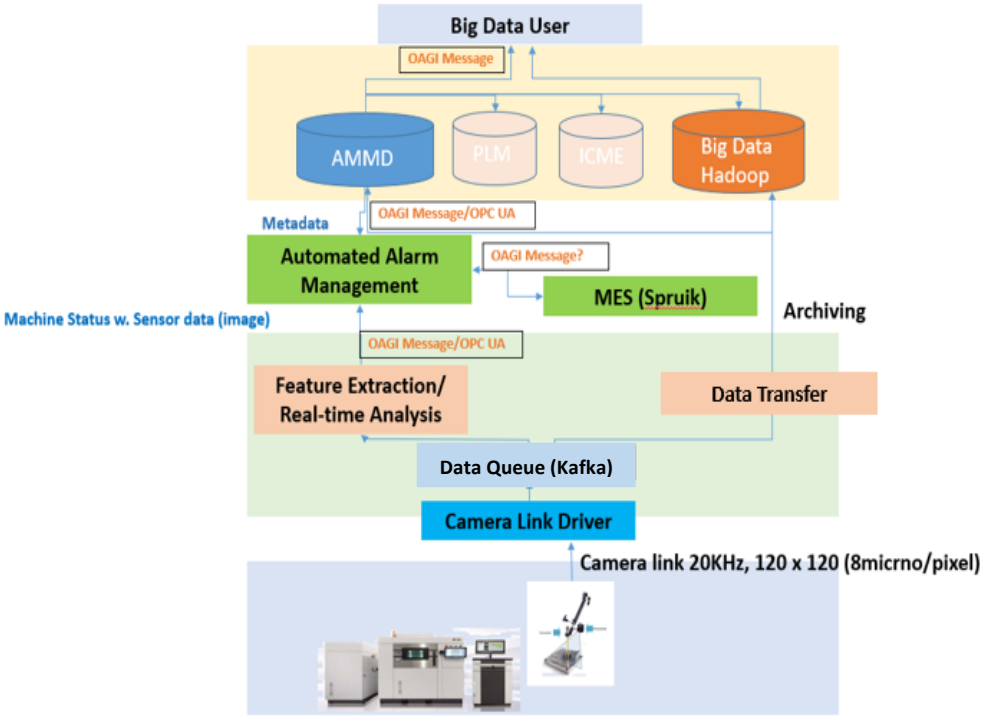


Figure 2. AM in-process Data Integration Architecture

**Collecting Data:** Data is collected using the Camera link protocol. The Camera Link Driver is responsible for collecting data from the sensors and forwarding them to the queue.

**Queueing:** The key to implementing this architecture is to have a buffer that will prevent the system overload, since the large amount of data must be processed in a short period. Apache Kafka[10], an open-source message queue, was successfully used for this purpose in our set up. Kafka can process many messages with negligible latency [8]. Also, Kafka is distributed, scalable system that offers high availability, so there is virtually no possibility of losing data.

Raw data from the camera-link driver is stored in a queue. To ensure that the system can work with the amount of data expected, it must be ensured, among other things, that the Kafka cluster can withstand that amount of data. The buffer can be considered as a single point of failure, which means that in case it does not work, the system will not respond. Therefore, it is of great importance that there is a reliable and redundant component in place such that the buffer is highly unlikely to fail.

[10] http://kafka.apache.org/

To ensure that Kafka can support the expected data flow, an overview of the capacity estimation is given below. In this use case, we assume that the smallest interval of change in an image is 50 microseconds. Each frame of the camera is 120 by 120 in grayscale; hence, the image file size is approximately 15KB. Considering that, 20000 * 15KB = 300 MB of traffic is expected in one second. LinkedIn engineering has published a benchmark for a three-node Kafka cluster with a simple configuration on servers with 7200 RPM SATA drives and 32 GB RAM, which are connected via 1Gb Ethernet. The benchmark confirms that traffic of 800K records per second can be processed by a cluster configured in this way without any anomalies in performance. The process was monitored until the traffic grew to 1400GB [7]. In the use case presented in this paper, the same server configuration was used. The data would be stored in a buffer for the next 24 hours from the time it is received to ensure easy system recovery in case of overload of any of the following components. Considering one hour of high traffic, the calculation is 300MB * 60 seconds * 60 minutes ~ 1100 GB, which is within the range that a cluster configured in this way can process, based on the LinkedIn's benchmark (1400 GB). During longer use of this system, the cluster can be further configured to be even more suitable for this use case.

From the queue, the images are sent for real-time analysis and for long-term archiving in parallel. We used Data Transfer, a custom application for preprocessing the data, for processing images and their metadata. In this way, we avoid storing irrelevant data in persistent storage.

**Archiving:** Different types of images are involved in this use case. HDFS, the file system of the Apache Hadoop, was used for storing them. Their metadata, however, were stored in Additive Manufacturing Materials Database (AMMD)[11], built on top of MongoDB. HDFS is responsible for storing large, structured, and unstructured datasets across various nodes. Data arrives frequently so it is not advisable to archive it in real-time. That could overwhelm the HDFS, because it is not designed to work well with many small files. For that reason, Sequence files are used that merge many images in one file. Approximately, 8739 images should be merged so the size of one sequence file corresponds to the size of the minimal HDFS block, which is 128MB. In this way, the HDFS block will be fully used. Also, queuing from the previous step will create a time delay to transfer the data to permanent storage less frequently and avoid an overload. The authors in [6] suggested MapFiles as another solution that could be used for storing images, but further research is needed before using this in real use cases.

AMMD is built using the NIST Material Data Curation System (MDCS) as a backend with structure provided by NIST's AM schema. Providing a collaboration platform, AMMD is set to evolve through open data access and material data sharing among the AM community.

**Real-time Analysis:** Next to being archived for future use, images and corresponding metadata are also used for real-time or near real-time analysis. Real-time functions include measurement data preprocessing - cleaning, melt pool feature extraction, anomaly detection, real-time feedback control generation, and emergency reaction, such as build stop. Near real-time functions fuse the melt pool monitoring data obtained from previous layers, conduct layer-wise process and part state evaluation, and make decision if the build should be continued or stopped. The real-time and near real-time analysis generate events, alerts and alarms which are sent to the Alarm Management system for operator to perform reasoning and take actions. An automated alarm management system can be built to replace humans in conducting the cognitive task by querying additional data from other systems, for example, the metadata data store, MES or ERP system.

[11] https://ammd.nist.gov/

**3.1.3 Data Flows:** In this architecture, there are several data flows that need to be defined. They are (1) between sensor and driver; (2) driver and Kafka; (3) Kafka and Data Transfer; (4) Data Transfer and Big Data Hadoop (5) Data Transfer and AMMD; (6) Kafka and feature extraction; (7) feature extraction and automated alarm management; (8) AMMD and automated alarm management; (9) AMMD and Big Data user.

For 7-9, we propose OAGI Message models or OPC UA standards, while the question remains which standards should be used for the 1-6 data flows. This requires further research because it is necessary to consider the technologies that have been proposed and their possibilities. Figure 3 shows a proposed high-level OAGI message for Data Flow marked with 9, in a typical BOD structure. A new noun named "AdditiveManufacturingBigDataPackage" is created to capture both the metadata and the reference links to the big datasets [4].

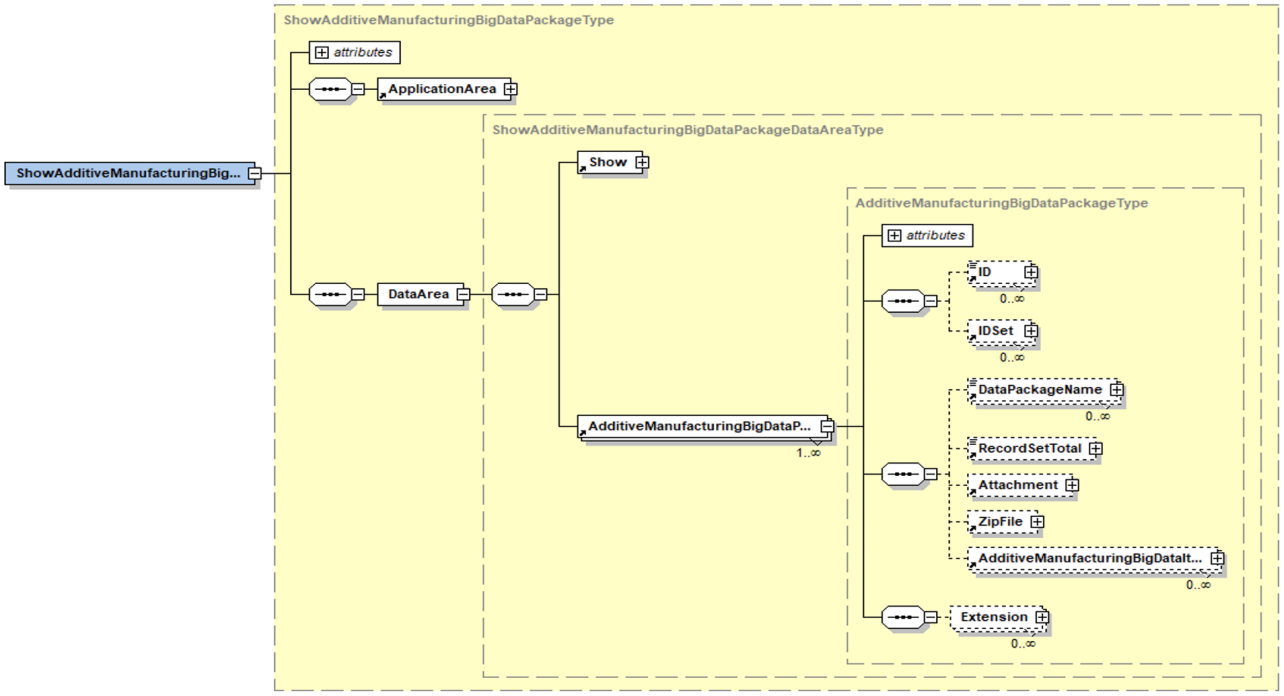


Figure 3. Additive Manufacturing Big Data Package OAGi message

## 4. Discussion

The AM data integration presents a new application of the five-layer ISA 95 architecture[12]. Layer 0 involves the functions and standards associated with the physical production process. Layer 1 involves the functions and standards associated with sensing and manipulating that physical process. Layer 2 involves the functions and standards associated with automatically monitoring and controlling that process. Layer 3 has manufacturing operation functions and Layer 4 are enterprise functions.

[12] https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95

### 4.1 Architecture Alignment

AM data integration involves data flows for both monitoring and control and streaming for big data archiving. Yet, we are mapping our architecture in Figure 2 to the lowest layers of the ISA architecture.

Alarm management provides Layer 2 functions we use for "controlling the state of the AM process". In addition, data integration for alarm management involves 1) the functions and standards associated with the physical production equipment (Layer 0) and 2) the functions and standards associated with sensing and manipulating that physical equipment (Layer 1). So, our AM data integration is functionally aligned with the ISA architecture for the branch of monitoring and control.

At the same time, data flows are no longer restricted - in both time and space - to adjacent, functional layers defined in ISA 95. For example, in-process data can be directly streamed and integrated with Layer 3 and Layer 4 functions. This data then becomes available to be analyzed and used for real-time decision making as well. Together with other enterprise data, proactive actions can be taken to optimize enterprise performance and improve the manufacturing strategies. Even for the lower-level integration, the interfaces will be different. They will contain new types of information and they will require new types of data structures; or, develop new ways of using or extending existing structures. We chose the latter. On-going discussions with both OAGi and MTConnect are focused on extending their message structures, creating new ones, and using/sending AM data in that resulting structure. The discussions identified a list of required messages, their content, and their senders/receivers. Technical discussions were conducted with a focus on finding technical approaches and solutions to meet those requirements.

### 4.2 Information Exchange Standards

Representative standards need to enable information integration and software interoperability both within and across each layer in Figure 2. Today, MTConnect and OPC UA[13] are the two major standards adopted by AM machine vendors for machine data integration. The MTConnect standard offers a semantic vocabulary and data models to grab data from machine tools, production equipment and other factory hardware. Data from shop floor devices is presented in XML format, and is retrieved from information providers, called Agents, using HTTP as the underlying transport protocol. OPC UA is a two-way communication protocol enabling either sending request and response messages between clients and servers or a publish-subscribe model.

In addition, a wide variety of Internet-of-Things connection options are emerging which might be applied to AM in-process data integration [5]. For instance, MQTT is a publish/subscribe protocol with minimal overhead and reliable communications, good for supervisory control and data acquisition (SCADA) and remote networks. Constrained Application Protocol (CoAP) provides the interoperability of HTTP but with minimal overhead, an appropriate substitution for edge-based devices where HTTP would be too resource intensive. Digital Data Service (DDS) is an open publish/subscribe protocol for fast and decentralized communication, best for machine-to-machine (M2M) communications.

[13] https://opcfoundation.org/about/opc-technologies/opc-ua/

All the communication protocols mentioned above provide an effective mechanism to integrate small volume data sampled at rates needed in traditional manufacturing processes. For the high velocity, high volume data generated from field devices, new standards are needed to integrate new types of data such as images. Given the complexity of the architecture and its new technology components required for AM data integration shown in Figure 2, existing standards must be re-evaluated and improved for smart AM.

## 5. Summary

This paper offers a comprehensive framework for data integration in additive manufacturing and, moreover, shows the practical implementation of the framework on the example of a melt pool monitoring system. In the paper, the key components of the framework are highlighted and an overview of the main decisions that must be made for the proper use of each step of the framework is given. Moreover, the proposed framework can be applied to other additive manufacturing systems and quickly achieve the desired results whether the goal is to archive data for future use or real-time analytics. The paper points that, while standards for the higher-level processing data flows are being developed already, there is a current lack of standards for messages exchanged in preprocessing data flows. Further research should include defining messaging standards for each of the observed flows.

## Disclaimer

Certain commercial systems are identified in this paper. Such identification does not imply recommendation or endorsement by NIST; nor does it imply that the products identified are necessarily the best available for the purpose. Further, any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NIST or any other supporting U.S. government or corporate organizations.

## References


1. K. Morris, Y. Lu, and S. Frechette, "Foundations of Information Governance for Smart Manufacturing," *Smart and Sustainable Manufacturing Systems* 4, no. 2 (2020): 43-61.
2. P. Boulware, “In-Process Monitoring Techniques for Laser Powder Bed Fusion”, https://ewi.org/wp-content/uploads/2018/12/LP4-Boulware_In-Process-Monitoring-Techniques-for-Laser-Powder-Bed-Fusion.pdf, Accessed on June 28, 2021
3. Y. Lu, Z. Yang and J. Kim, “Camera-Based Coaxial Melt Pool Monitoring Data Registration For Laser Powder Bed Fusion Additive Manufacturing”, Proceedings of The International Mechanical Engineering Congress and Exposition, 2020
4. Y. Lu, “AM Big Data Registration and Exchange based on OAGIS Message Modeling”, ASTM International Conference on Additive Manufacturing (ASTM ICAM 2020)
5. Y. Lu, A. Jones and P. Witherell, “Standard connections for IIoT empowered smart manufacturing”, Manufacturing Letters, Volume 26, October 2020, Pages 17-20
6. Q. Su,, L. Luand Q. Feng, ”An optimal solution of storing and processing small image files on Hadoop”, In *International conference on brain inspired cognitive systems* (pp. 644-653). Springer, Cham, 2018
7. J. Kreps, “Benchmarking apache kafka: 2 million writes per second (on three cheap machines)”. Online: https://engineering. linkedin. com/kafka/benchmarking-apachekafka-2-million-writes-second-three-cheap-machines, Accessed on June 28, 2021

8. A. Warski, “Using Kafka as a message queue”. Online: https://softwaremill.com/using-kafka-as-a-message-queue/, Accessed on June 28, 2021.