A Gap Analysis and Improvement of Additive Manufacturing Data Interoperability Models

This project focuses on information modeling in additive manufacturing (AM), specifically laser powder bed fusion. Such additive processes both generate and use a significant amount of data. Standardized information models have the purpose of organizing this data in such a way that it is findable, accessible, interoperable, and reusable (FAIR). The two data models explored are the Common Data Dictionary (CDD) and the Common Data Model (CDM). The CDD defines attributes of data (ie. name, definition, and data type) featured in the AM ecosystem in an excel spreadsheet. The CDM is the standardized organizational format for AM data, expressed in an XSD schema.

The work addresses one major problem: these two data models, the CDD and CDM, are not compatible with each other.

The goal of the project is to create an algorithm that detects and documents discrepancies in data attributes between the standardized CDD and CDM under development for the improvement of interoperability.

This goal was approached with various methods.

The first method is to conduct manual data mapping. Data mapping is the process of matching data fields across different data formats, simultaneously capturing differences between the data formats. By manually mapping the 'build' information module in the CDD to the 'buildType' class in the CDM, knowledge and logical understanding of the methods and conditions for matching were generated. This manual process is assisted by the software, OxygenXML. Following this exercise, matches between data elements can be categorized into four cases: having matching names, having matching definitions, having no exact matching attributes but matching meaning, and having no matching element at all. Where an attribute is inconsistent between data formats, the attributes in the CDM are updated with the attributes defined in the CDD. Different classes in the CDM have different distributions of these match cases. While this manual mapping process is accurate, it is time consuming.

The second method is to conduct automatic data mapping in order to increase the efficiency of the process. This was done so by developing a python code involving pandas data format and JSON text format. The abstract workflow is as follows: The CDM XSD file and CDD excel file are parsed into JSON text format and a pandas dataframe, respectively. The CDM is then iterated through once. For each element in the CDM, the entire CDD is iterated through once in search of a match. If a match is found, the attributes of the CDM element are updated with the attributes of the matching CDD element. If no match is found, the loop continues onto the next CDM element. When all CDM elements are iterated through once, the loop ends and a revised CDM is returned in JSON text format. While it would be straightforward to check for matches based on the four categories of match cases outlined above, the computer is unable to detect matches that fall under the category of having no exact matching attributes but matching meaning.

The third method is to incorporate language processing into automatic data mapping. Such language processing intends to establish a common concept system between the CDD and CDM so that matching can be conducted by solely comparing data element names. This is achieved by filtering out “stop-words” from data element names. When it comes to determining which words will be “stop-words”, the frequency of each word amongst all words that compose data element names in the CDD is considered. Words with a higher frequency are correlated with less semantic value since they are featured in a multitude of data element names, and therefore do not play a key role in differentiating one element from another. Such words are more effective as “stop-words”. In contrast, words with a lower frequency are correlated with more semantic value since they are featured in few data element names, and therefore do play a key role in differentiating one element from another. Such words are less effective as “stop-words”. Based on these frequencies, three thresholds of “stop-words” representing the percentage of all individual words in all CDD element names composed by a unique word are established: ≥1.00%, ≥2.00%, and ≥3.00%. After understanding how to determine these “stop-words”, the data element names are processed by removing “stop-words”. This is first done manually for the ‘build’ module in the CDD, determining and removing “stop-words” based on human judgment and background knowledge. While the accuracy of manual processing is relatively higher, it is time consuming. This is then done automatically with a python algorithm for the entire CDD. The removal of “stop-words” in the automatic process has been tested with the three different thresholds, yielding different outcomes.

While significantly more efficient, this process encounters challenges dealing with inconsistent semantic value of words. A given word may be significant in some contexts and insignificant in others. Since such inconsistencies are not detectable by the algorithm, difficulties regarding appropriate instances to remove the given word arise. Problematically, removal of a given word when significant results in duplicate element names or empty strings. Duplicates indicate that a conjunction of words is significant, and empty strings indicate that individual words are necessary. At this point, the processed element names must be returned to the original data element name, undermining the language processing strategy.

Two sets of results are obtained from these three methods.

The first set of results relates to the automated language processing and automated mapping method. After testing the removal of “stop-words” using three different thresholds, metadata on the number of unique duplicated names, individual instances of duplication, and empty strings is collected (prior to returning duplicates and empty strings to original element names). It is observed that as the threshold increases from ≥1.00% to ≥3.00%, the number of unique duplicated names, individual instances of duplication, and empty strings decrease steadily. Data on the number of successful matches made in the ‘buildType” class of the CDM under each threshold is then collected (after returning duplicates and empty strings to original element names). Interestingly, it is observed that the greatest number of matches falls under the middle threshold of ≥2.00%. This indicates two critical findings: 1) reducing the number of duplicates and empty strings does not necessarily relate to increasing the number of successful matches made and the effectiveness of automatic mapping, 2) An ideal “stop-word” threshold for automated element processing exists and this process has the potential to be optimized.

The second set of results relates to the overall success in data mapping across the three approaches explored: 1) manual mapping with no language processing, 2) automatic mapping with manual language processing, 3) automatic mapping with automatic language processing. Results of mapping are labeled as true positive, false positive, true negative, and false negative depending on whether or not a match exists and whether or not a match is detected. Having been reviewed by expert sources, the first manual method has an accuracy of 100% (77.8% true positive, 22.2% true negative) and serves as a baseline to compare the success of the other two methods. The second method has an accuracy of 77.8% (55.6% true positive, 22.2% true negative, 22.2% false negative). The third method has an accuracy of 72.2% (50.0% true positive, 22.2% true negative, 27.8% false negative). It is observed that as the level of automation increases in the mapping process, the percentage of true results decreases and the percentage of false results increases. However, it is also observed that the percentage of true negatives remains constant throughout and no false positives result at all. This indicates that the algorithm is reliable in not detecting a match when it does not exist. In terms of further development, efforts can be concentrated on reducing the appearance of false negatives.

By investigating these methods, it can be concluded that semantic mapping via language processing increases the efficiency of automated data mapping. However, there are current challenges with harmonization that exist and cannot be overlooked: 1) the existence of different naming conventions across different data formats, 2) errors in data element inputs (e.g. spelling errors, classification errors). In the future, further development of standards and software to overcome these challenges with harmonization will enable seamless integration. In expanding applications, “stop-word” libraries can be implemented into natural language processing tools for additive manufacturing. It may be valuable to apply similar concepts to fields that involve high data flow outside of additive manufacturing as well.

Generative AI is an emerging tool for software development. While it is still undergoing development and poses challenges involving inaccurate or insufficient output, it opens the door for learning and is a resource that is highly valuable in the future.