Evaluating four types of data parsing methods for machine learning integration from building information models

Chapter · March 2023

DOI: 10.1201/9781003354222-14

CITATIONS
0

READS
24

4 authors, including:

Povl Filip Sonne-Frederiksen
Arkitektskolen Aarhus
5 PUBLICATIONS **0** CITATIONS
SEE PROFILE

Peter Nørkjær Gade
University College Nordjylland
23 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Evaluating Four Types of Data Parsing Methods for Machine Learning Integration from Building Information Models

F. Sellberg & J. Buthke & P. F. Sonne-Frederiksen

Link Arkitektur, Stockholm, Aarhus, Copenhagen, Sweden, Denmark

P. N. Gade

University College of Northern Denmark, Aalborg, Denmark

A method and structure for architectural datasets specifically designed for the analysis, sorting, and ultimately reusing of building elements is proposed. Four different methods of parsing data from real-life projects using their building information models (BIM) for integration into a machine learning (ML) model were evaluated. As ML integration is becoming more important in the Architectural Engineering and Construction (AEC) industry, we see an increasing demand for high quality datasets. Four different methods and file formats were benchmarked, focusing on read and write-speeds for converting architectural BIM into datasets to be used in ML. Our results show that the current way of storing our projects in Industry Foundation Classes (IFC) is not optimal for the development and integration of new Artificial Intelligence (AI) assisted tools. This paper provides alternative methods and storage solutions for both developing new datasets internally and also for future work in creating a common federated learning setting for the AEC industry.

Introduction

The construction industry is becoming increasingly complex with more and changing requirements with regard to climate change and sustainable needs. These new requirements further necessitate that the information created and exchanged is of high quality with regard to reliability and consistency. Moreover, the information required to create and manage building designs is continually increasing and requires that it can be accessed and edited by that user quickly.

A major problem related to managing information for building projects is the time spent by designers in managing information. A study by Flager et al. (2007) found that designers, in general, spend about half of their time managing information. Hereby, designers are not spending their time creating new design information or doing analysis. Instead, the designers are often burdened by merely moving information around to ensure consistency and quality, i.e., coordinating existing information.

In order to reduce the time designers, spend on managing information, new methods like Artificial Intelligence (AI) can be applied to automate this process, making it more efficient and ensuring high-quality information across the building projects (Song et al., 2018; Zabin et al., 2022). While AI is noticed in many larger construction companies, it is still considered a fringe technology that is slowly being implemented in the industry (Abioye et al., 2021; Molio, 2020; Natonal BIM Standards, 2020).

One of the major barriers to implementation is that the technology, in many aspects, is immature and requires much skill to apply. As Abioye et al. (2021) argue, new roles in the construction industry need to be formed in order to cater to utilizing the benefits of AI. Moreover, acquiring these skills is also a major barrier due to the general talent shortage of people with skills in AI.

Kyicska & Tsiutsiura (2021) argue that in order to better make use of AI in the construction industry, there needs to be a better understanding of what the users need. This can be done by experimenting with AI in order to identify potential solutions of using AI for relevant cases for the industry. Specifically, machine learning (ML), a concrete AI methodology, has been promising for various use-cases in the construction industry to solve various issues by using learning data to train algorithms to make predictions that can be used in making decisions.

Data model

Open datasets for use in the AEC industry are scarce and often consist of projects in a very isolated scope of context. The few high-quality datasets that have been made are often made for a large urban scale. Lu et al. (Lu et al., 2019) showed an approach using convolutional neural

network where 10.000 images was used from a case named 5M-Building as a dataset to detect buildings in pictures. However, such an approach is difficult to implement in a more common architectural scale. To tackle this issue, we propose that datasets can be created from a resource that most firms already have, their existing projects and building information models (BIM).

In our methods of converting from a BIM, we evaluate what file storage is relevant and how they can be used for a ML context. In an article by Wang & Tang (2021) it was suggested to save BIM information based on IFC into databases for long term storage using Java language and MySQL database. Based on that they created a prototype called IFCParser. Creating the prototype, they found that it assisted engineers that weren't knowledgeable about IFC to easier to get and store BIM information on their own servers to focus on solving problems.

Withers (2022) discuss in that a global shift is happening where assets move from tangible to intangible. Through new ways of storage, a more easily sold and quantifiable asset can be created for architectural firms in the way of filtered project data.

In order to find a more optimized and stable file format for both long term storage and fast integration to new development of tools with a special emphasis on development of Machine learning methods on large architectural data sets. Easing designer's workload from managing complex data parsing from building information format to another. This can be done by automating this work by using ML.

In this article we measured four different file formats with our key metrics for evaluation, which are file size, write-speeds from native BIM software formats to a new file format, and read-speeds when loading it into a ML model to showcase the potential advantages of automating data parsing across the different showcased filetypes. These insights can potentially highlight the different approaches to using ML for data parsing in the AEC industry to help alleviate current challenges.

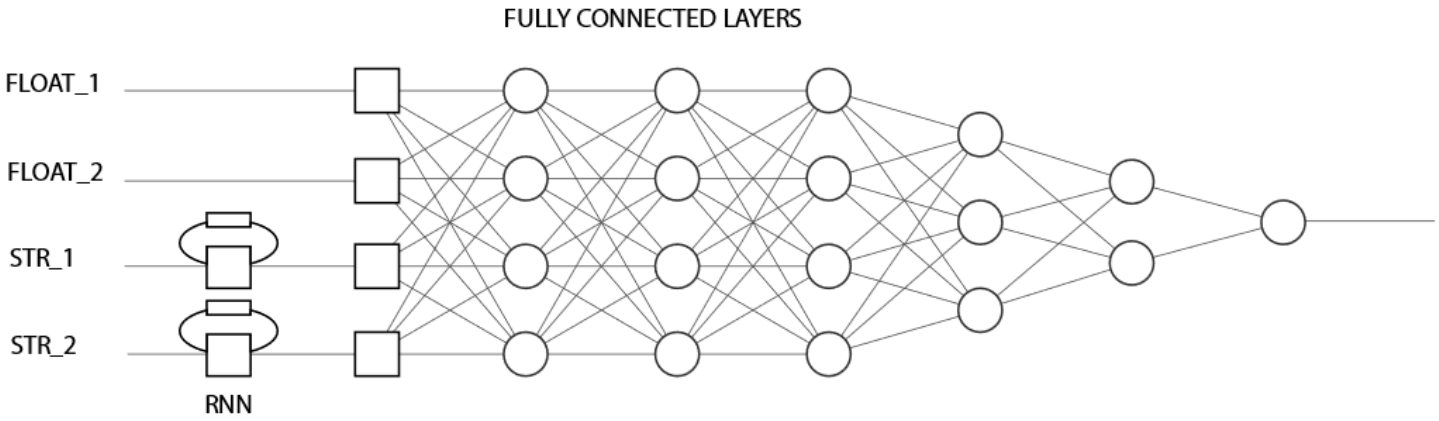

Figure 1. Neural network prototype with Recurrent Neural Networks (RNN).

Methodology

Our prototype is developed and evaluated using information from real-life building projects in Sweden and Denmark. Such an investigation can give insights into practical applications of ML in improving the handling of building project information stored in BIM.

The proposed method for evaluation consists of two steps; the first is exporting a BIM into four commonly used file formats. The file formats were picked from common architectural use or in data science. In this process, we are evaluating the write time and file size. As a second step, we are loading the new exported files into a ML model consisting of a recurrent neural network and evaluating the read time of our data set used in the training of the neural network for each storage solution.

File formats

Our four file formats that we evaluate are:

Industry Foundation Classes (IFC4)

IFC4 is the most used file-format for interoperability and long-term storage of projects in architectural practices today. IFC is an object-based file-format for description of architectural data. The data structure is formatted to easily be read by a multitude of software through the creation of standardized element definitions.

Json (2020-12)

Json is a lightweight open standard file format using attribute value pairs, often used in transmitting data in web applications. It is not the most widely used format for machine learning but was chosen for its high human readability, flexibility, and ease of implementation.

Speckle (v2)

Speckle is an open-source cloud-solution for BIM. It is used to stream projects onto a cloud server for interoperability and can be used as a long-term cloud-storage. It uses a similar structure to an IFC; for this specific format, we only evaluate write and read-speeds with no size comparison since a cloud-based database has no project specific file size related to it.

Petastorm (0.11.2)

Petastorm is an open-source data access library using Apache Parquet datasets originally used for real-time deep learning for self-driving vehicles. It uses many modern approaches to optimize high velocity data feeding, as described by Qiu & Sun (2015), into a ML model such as local caching and sharding.

Neural network prototype

The ML model is dependent on the individual use case. Here, for this example, we chose to train the model to predict CO_2 emissions of different building elements. For that, we are feeding the model with dimensional information (width and volume) and information about object name and material. This means that we are dealing with two different kinds of data. On the one hand, we have number values, and on the other, we have 'strings' (text). Because an ML model can only handle numerical values, the text first needs to be converted and processed. This is done by converting each character into a one-hot encoded vector and then parsing those vectors through a recurrent neural network (RNN) to get a single classification of that string (see Figure 1).

This classification is then used together with the other dimensional values to calculate a final CO_2 emission by parsing those values through a fully connected neural net.

Prototype evaluation

We are evaluating the prototype according to the following variables: write-speeds from the BIM to a storing format; storage sizes of each format and how they can be stored; read-speeds from each format into a ML model; and test run each to see how they perform with respect to time.

To evaluate our method, we chose five larger projects in Denmark and Sweden that are all in the process of being or have been constructed and have a focus on sustainability. Our selection consists of residential projects, as they contain more elements to sort our evaluation and have more complex interior structures (see Table 1 for more detailed project information).

NAME	LOCATION	YEAR	FUNCTION	SQUAREMETERS	SOFTWARE
LIDL VIGGBY	Sweden, Stockholm	2020	Commercial	4323 m^2	Archicad
LOJOBACKEN	Sweden, Stockholm	2022	Residential	8045 m^2	Archicad
FRIPLEJEHJEM	Denmark, Haderslev	2020	Residential	6633 m^2	Revit
STRANDBOLIGERNE	Denmark, Copenhagen	2021	Residential	3284 m^2	Revit
TV BYEN	Denmark, Copenhagen	2022	Residential	10312 m^2	Revit

Table 1: Selected architectural projects and data.

Results

Write-speeds

Write-speed is the time (measured in seconds) that it takes to export from Revit into the specified file format (see Figure 2). For Json and Petastorm, only walls were exported, as our ML model was being trained on predicting kg/CO_2 for walls. For IFC and speckle, all elements were exported, as is the standard of the formats. To simplify the comparison of the different formats, a graph of write-speeds in relation to the output file size was created to see how fast each method breaks out the relevant information. Something to notice is the size of the file created for each relevant dataset. Petastorm is the smallest, with an average of 0.026 MB, and IFC is the largest, with an average of 101.962 MB. As speckle is using a cloud storage solution, it is excluded from the time/size comparison graphs.

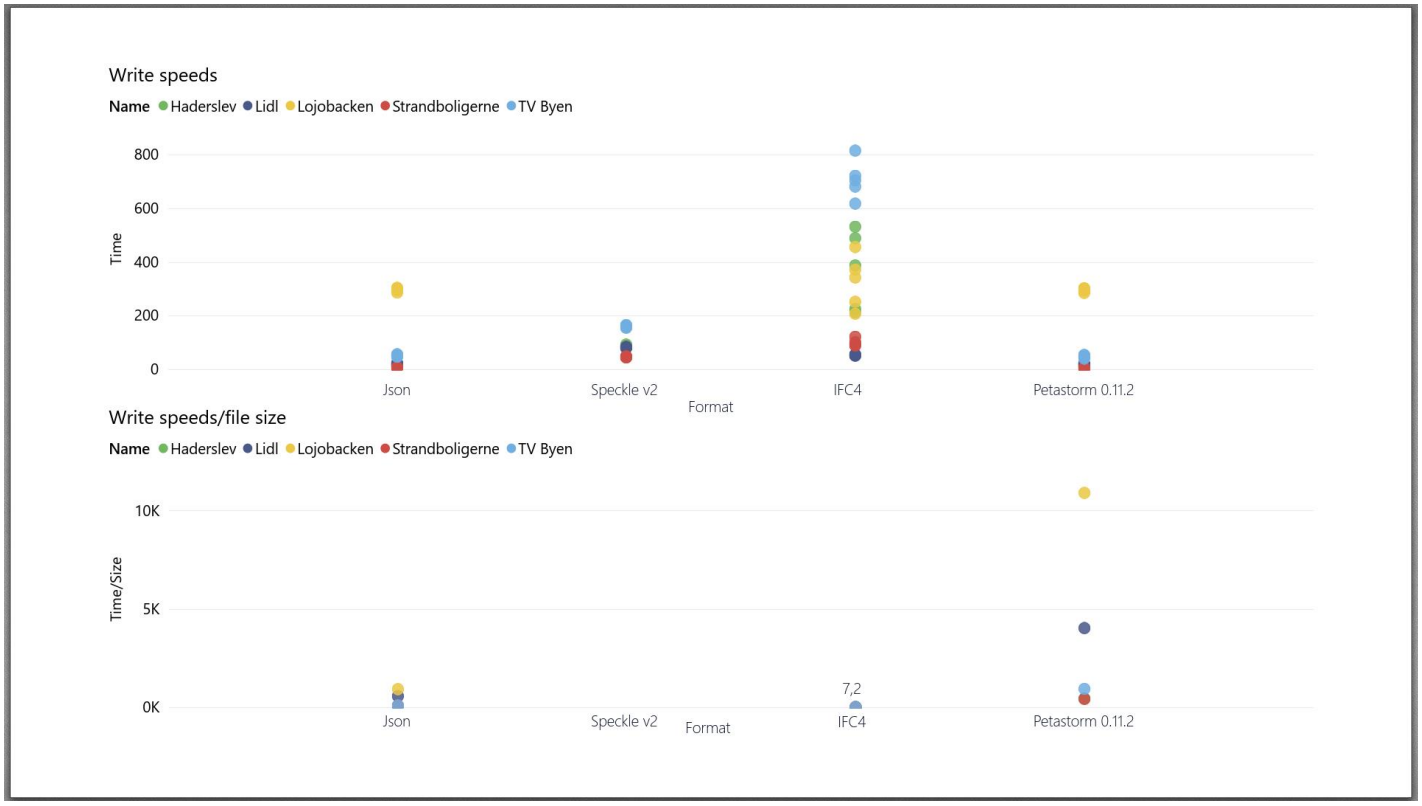

Figure 2: Write-speeds and write-speeds by file size for the formats.

IFC4

IFC, with an average write time of 310.9 s, has the longest write time of all the file formats in the comparison. This is largely because in our evaluation method, we exported all elements in our projects. When looking at write time/size, we can see that it performs the best. Because it is a file format made for interoperability, optimized for exporting large and complex projects, as well as being largely supported by BIM software, this was to be expected.

Json (2020-12)

Json, with an average write time of 77.8 s and an average file size of 0.249 mb, provides architects with a fast method to export smaller segments of relevant information in architectural projects. When comparing write time/size, Json is the slowest in filtering out and storing relevant elements from the BIM.

One would expect slightly faster speeds, as the format is made and optimized towards data interchange, but as the format provides a more human-readable text than the other formats, some overhead in the files is created, which shows in the write-speeds.

Speckle (v2)

Speckle has the fastest average write time of 74.7 s with no apparent file size because it uses a cloud solution for saving the files, which obfuscates the file sizes. Something to note here is that the largest project “Lojobacken” was not correctly exported, so the longest write times are missing. This shows the main weakness in the system, i.e., the BIM projects need to be of a high quality to be able to be uploaded to Speckle.

Write times include time to upload the project to the cloud, so it is heavily dependent on network speeds. In a similar fashion to how we treat IFCs, we are evaluating write times of an entire project. With write-speeds being significantly faster than an IFC export and keeping all relevant information for both analysis and interoperability, this format is very promising.

Petastorm (0.11.2)

Petastorm results in write times that look very similar to the results from Json, as they share a similar code base for filtering and exporting from BIM. Something to notice here is the slightly faster write-speeds and the significantly smaller file sizes, as a parquet-based database is being utilized instead. Write time/size is slightly skewed, as file sizes are so small in relation to the amount of information in them.

Figure 3: Read-speeds and read-speeds by file size for the formats.

Read-speeds

Read-speed is the time (measured in seconds) that a ML model takes to read and import the data from the different formats (see Figure 3). For IFC and Speckle, a filtering of walls had to be made, while Json and Petastorm already were filtered to only include walls in exporting to the file formats.

To help compare the different formats that use a slightly different method of importing, a graph of read-speeds in relation to the input file size was created. As Speckle is using a cloud-based storage solution, it was excluded from the read time/size results.

IFC4

The average read time of IFC is 7.0 s, which is very fast considering their large file sizes. As the file system is made to be able to quickly import large files between software, this could be expected. A conflicting relationship can be seen through the results in that the larger the number of unique elements in the file gets, the read-time becomes exponentially longer. For large scale projects or aggregated projects, this will become an issue as the read times increase dramatically.

Json (2020-12)

Json has an average read time of 0.004 s and very small file sizes, as it only has the already filtered information in them. The lowest read time/size can still be seen. No decreasing speeds in relation to an increase in the number of elements in the files can be seen. For larger projects, Json would be the preferred file format.

Speckle (v2)

Speckle provides the slowest read times at an average of 24.8 s, while this largely depends on the network connection. A relation to write times can be made where Speckle performed significantly better than IFC exports. No file size exists to compare against, but the number of elements in both IFC and Speckle are the same, as shown in Figure 3. IFC is performing more than 4 times faster than Speckle in filtering and importing the elements.

Petastorm (0.11.2)

Petastorm has an average read time of 0.096 s, which is slightly slower than Json. This is mostly because it must convert and load the data into a structured spark data frame. When comparing read-speeds to file sizes, we can see a decrease in the evaluation metric the larger the project is. This is because the slowest process in the

method is in creation of the data frame itself which the larger the file gets becomes a smaller process.

So, for larger projects, a faster value in regard to read-speeds/size can be expected. Many of the overhead functions provided by the format, such as real time updates and sharding, are not used but are functions that can be used to heavily improve dataset processing in very large data sets.

Discussion & Conclusion

This research plays a crucial role, as it constitutes a part of the conceptual basis for a new way to build up large datasets optimized for ML algorithms to read and write faster. The digitalization of many processes in the AEC industry is increasing, but very large unstructured datasets are very common, such as the dataset on five million buildings (Lu et al., 2019).

To be able to fully leverage those datasets, the industry requires fast ways to search through the datasets and extract the specific information that is needed. One field of application is the reuse of building parts between building projects, where specific reusable building parts could be found and matched between datasets (1. Representing the building to be erected 2. Existing building acting as material banks). This process could potentially support a more circular future in AEC.

Four methods were developed for the export and import of five real life projects into four different file formats, which then were loaded into a ML model predicting kg/CO_2 in wall elements. Here, we outline the possible use case scenarios of each format for the AEC industry.

IFC current usage scenarios of full project interoperability and long-term storage with 3D information might not be its best use cases. With long write times but fast read times, the format is better suited for importing information into the analysis of complete projects such as ML models trained on tagging untagged building elements.

A problem with IFC files is that they include very detailed data for commercial data purposes, where a more anonymized approach is required. For development of new tools where data from multiple projects are required, another file format should be applied.

Petastorm usage is great for very large, aggregated datasets for ML with a predetermined function. With each element in the data frame needing to be pre-defined and converted into a tensor, a large technical understanding is needed to set up the exported data to be able to be both exported and imported in a correct way.

Database approaches are great for small file sizes; this further supports the use in very large datasets, which is not something that is common in the AEC industry. Being a newly and niche developed system, integration into common software and libraries is not fully extended, giving it a clear disadvantage for developing tools connected to the format.

Json has several use cases; in a ML context, its best application would be in aggregating filtered parts of projects where its small file size and fast read and write-speeds can be utilized. Examples would be in predicting wall material compositions or CO_2 emissions. Another clear use case is in tool development where its common use in the field is a large contributing factor.

Large shares of existing software and libraries already provide integration so development time can be decreased. Because data are highly adaptable and easy to make anonymous, Json would be the preferred format for commercializing project data for selling/buying between firms.

Speckle provides interesting use cases where a fast write-speed but slower read-speeds gives it good use cases ranging from interoperability to the creation of non-platform specific tools. For inoperability and long-term cloud storage, the format excels, where read times are not as important and the flexibility of the format is more prioritized.

A note is that the format is dependent on externally developed connectors for importing and exporting data, so a more optimized filtering and reading can be developed to further decrease read times. It has a low barrier of entry in the use and development, though a higher technical understanding is needed when optimizing and developing its connectors.

Hawkins (2020) proposed a method for finding the minimum viable model and how many ML projects start out without a viable Return on Investment. A similar discussion can be had on architectural projects, where many firms will not be able to find enough data to support ML trainings.

An architectural project usually contains a great deal of information in different elements; the problem is thus the lack of quantitative data on specific element parts. While we evaluate how the different formats perform in speeds/size to find a clear comparison number for evaluation, a real dataset with thousands of projects would perform very differently. This is something the entire industry will have to tackle, as the sheer number of pro-

jects needed to perform a prediction based on a single element would be beyond what one firm could muster.

As the AEC industry is moving forward, there is another major problem that comes up, which is non-contextual datasets either from datasets not relevant to the current context or from computer generated datasets. This is an issue that has been growing in recent years.

The AEC industry is extra vulnerable to non-contextualized data, as building laws and standards are high variable between countries and continents. As larger datasets become more varied and lacking in specific data, a high number of local datasets would have to be created. This can be solved through just further training a model, although a lack of labeled data in the AEC industry would make this difficult.

One option to handle the lack of data could be to investigate methods of federated learning setting outlined in Kairouz & McMahan (2021) where each party just trains the model on the portion of data that they have without revealing it. Such approaches are being developed, for example, in the medical industry where patient data are highly confidential.

The drawback though is that it does not incentivize those with substantial datasets to participate, as they do not stand to gain as much compared to their contribution. Furthermore, those methods rely on the honesty of all participants to not corrupt the model by feeding it wrong information.

Another angle for future investigation would be to investigate other forms of data. For example, not all projects exist as BIM from which data can be extracted. Some only exist as drawings or in their final build form. Being able to extract information from other datatypes would open possibilities for different use cases and expand the pool of available data immensely.

One such case would be regarding existing structures and their transformation when it comes to circular economy. Being able to process point cloud scans, a simple method of digitizing buildings efficiently would enable the harnessing of the information embedded within those as well as information that has been aggregated over the lifetime of the building. Opening future use cases, for example, when it comes to the repurposing and transformation of those buildings.

References

Abioye, S.O., Oyedele, L.O., Akanbi, L., Ajayi, A., Davila Delgado, J.M., Bilal, M., Akinade, O.O., et al. (2021), "Artificial intelligence in the construction industry: A review of present status, opportunities and future challenges", *Journal of Building Engineering*, Elsevier Ltd, Vol. 44 No. October, p. 103299.

Flager, F. and Haymaker, J. (2007), "A comparison of multidisciplinary design, analysis and optimization processes in the building construction and aerospace industries", *24th W78 Conference on Bringing ITC Knowledge to Work*, pp. 625–630.

Hawkins, J. (2020), "Minimum Viable Model Estimates for Machine Learning Projects", pp. 37–46.

Kairouz, P. and McMahan, B. (2021), "Advances and Open Problems in Federated Learning", *Advances and Open Problems in Federated Learning. Foundations and Trends® in Machine Learning*, pp. 1–210.

Kyivska, K. and Tsiutsiura, S. (2021), "Implementation of artificial intelligence in the construction industry and analysis of existing technologies", *Technology Audit and Production Reserves*, Vol. 2 No. 2(58), pp. 12–15.

Lu, Z., Xu, T., Liu, K., Liu, Z., Zhou, F. and Liu, Q. (2019), "5M-Building: A large-scale high-resolution building dataset with CNN based detection analysis", *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, IEEE, Portland, OR, USA, Vol. 2019-Novem, pp. 1385–1389.

Molio. (2020), "Byggeriets Digitale Barometer", No. September.

Natonal BIM Standards. (2020), "10th Annual UK's National Building Specification Report 2020", *NBS Enterprises Ltd.*, pp. 1–39.

Qiu, J. and Sun, Y. (2015), "A Research on Machine Learning Methods for Big Data Processing", No. June 2016, available at:https://doi.org/10.2991/icitmi-15.2015.155.

Song, J., Kim, J. and Lee, J.K. (2018), "NLP and deep learning-based analysis of building regulations to support automated rule checking system", *ISARC 2018 - 35th International Symposium on Automation and Robotics in Construction and International AEC/FM Hackathon: The Future of Building Things*, No. Isarc.

Wang, R. and Tang, Y. (2021), "Research on Parsing and Storage of BIM Information Based on IFC Standard", *IOP Conference Series: Earth and Environmental Science*, Vol. 643 No. 1, available at:https://doi.org/10.1088/1755-1315/643/1/012172.

Withers, L.W. (2022), "The accelerated shift to intangible assets and how to protect them", available at: https://global.lockton.com/gb/en/news-insights/the-accelerated-shift-to-intangible-assets-and-how-to-protect-them (accessed 7 April 2022).

Zabin, A., González, V.A., Zou, Y. and Amor, R. (2022), "Applications of machine learning to BIM: A systematic literature review", *Advanced*

Engineering Informatics, Vol. 51 No. April 2021, available at:https://doi.org/10.1016/j.aei.2021.101474.