An approach for data extraction, validation and correction using geometrical algorithms and Model View Definitions on building models

Luttun, Johan; Krijnen, T.F.

DOI
10.1007/978-3-030-51295-8_38

Publication date
2020

Document Version
Accepted author manuscript

Published in
Proceedings of the 18th International Conference on Computing in Civil and Building Engineering (ICCCBE 2020)

Citation (APA)
Luttun, J., & Krijnen, T. F. (2020). An approach for data extraction, validation and correction using geometrical algorithms and Model View Definitions on building models. In E. Toledo Santos, & S. Scheer (Eds.), *Proceedings of the 18th International Conference on Computing in Civil and Building Engineering (ICCCBE 2020)* (pp. 529-543). (Lecture Notes in Civil Engineering; Vol. 98). Springer. https://doi.org/10.1007/978-3-030-51295-8_38

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

An approach for data extraction, validation and correction using geometrical algorithms and Model View Definitions on building models

Johan Luttun[1] and Thomas Krijnen[2]

[1] AECgeeks, 5616TW Eindhoven, The Netherlands
[2] Delft University of Technology, 2628BL Delft, The Netherlands
lncs@springer.com

Abstract. The Industry Foundation Classes (IFC) cover a wide variety of subdomains in the construction industry. Model View Definitions (MVD) enable to specify a subset of the IFC schema to assess the content of a model for specific use cases and information exchanges.

However, IFC and MVD paradoxically complexify the workflow since it requires a deep understanding of the schema combined with construction knowledge to carry out simple use cases such as quantity checking or data export. This gap between domain specific queries and their expression in a computer-readable language weakens the opportunities provided to the building industry by Building Information Modeling.

Our research consists in the implementation of MVDs in a high-level programming language to extract data from building models, an assessment of the extraction results and geometrical processing algorithms to correct the explicit quantities and properties that are supplied as metadata alongside the elements in IFC building models.

Geometrical processing can be used to reduce and eventually correct errors on property values. We use a generic geometrical representation of IFC entity instances and apply geometrical transformations on those to obtain geometrical shapes. Boolean operations are used to identify relationships between elements. Eventually, incorrect data values are corrected directly in the IFC models accordingly to the IFC schema.

For instance, we authored an MVD to extract data pertaining to external walls from different IFC models and corrected the value of the IsExternal property of the models' IfcWall entities. This use case is of great importance for the cost estimation of a thermal renovation on a building as it gives a good estimate of the outer surface area of the building envelope.

Keywords: Building Information Modeling (BIM), Industry Foundation Classes (IFC), Model View Definitions (MVD).

1 Introduction

1.1 Industry Foundation Classes

The Industry Foundation Class (IFC) standard, developed by buildingSMART, a worldwide community of actors from the construction industry, aims to provide interoperability between software applications while describing specific domain-related products. Indeed, the density of information conveyed between the project actors and used by different software applications requires a neutral format for import and export. However, the IFC models used in the industry contain several errors [1] regarding to data accuracy, structure, or existence, which compromises the benefits brought by a BIM workflow: saving time and increasing project quality thanks to the storing of reliable data. The multiple translations between software applications, the heterogeneity of domain activities, and the diversity of use cases make digital exchanges error prone. In addition to data related errors, geometry errors are also responsible for unusable models and depreciating BIM added value.

Elements in a building element model have one or more geometrical representations, a placement that locates the element in 3D space and a set of associated semantic constructs such as material information and property sets, in which the latter allow for more user- or application defined key-value pairs. A subset of the property names in a model follows standardized patterns by buildingSMART, for example for IfcWall elements information, a field IsExternal is typically available to document whether the element is part of the external façade of the building. There are also numeric properties such as volume and surface area available in quantity sets.
The geometry definitions in IFC constitute a large part of the IFC schema and a large part of the considerable implementation effort for applications to fully support IFC. These constructs, called representation items, can be simple explicit polyhedra or fairly complex sweeps and revolutions with boolean operations applied to them. The implicit geometry definitions and hierarchical placements make discovering simple geometric relationships between elements such as containment and adjacency, computationally expensive. Therefore, these relationships such as spatial containment in the project decomposition structure, connectivity information between walls and space boundaries are explicitly provided in the model as objectified relationships.

One of the most investigated topics in the construction industry is the reduction of the energy consumption of buildings and the limitation of their environmental hazards, particularly the carbon footprint, at each phase of their whole life cycle [2]. Thanks to IFC specifications, forecasting the CO2 released by a building throughout these phases, which requires an accurate modeling of quantities and materials used, is more affordable. The data stored in an IFC file offers a real opportunity to carry out such use cases, that depend on specific and accurate data. For example, the capacity to extract information about the external walls to calculate the façade area of a building is an important step towards the appropriation of BIM by end-users because it can be incorporated into

decision-making processes, for instance at design or at earlier phases. The estimation of physical quantities allows for use cases that answer to other than purely technical rules, such as the financial ones. Thermal losses are determining variables for the calculation of energy consumption, which is a main part of a building's CO2 emissions [2]. Since conductive heat transfer takes place between heated and non-heated volumes through their common surface [3], insulating the building façade with less permissive materials reduces this transfer rate, and thus mitigates the amount of energy needed to maintain the thermal comfort temperature. In addition, façade geometry and materials intervene in the comfort, financial, and esthetic metrics of a building, and has been the object of optimization research[3, 4]. Therefore, using IFC to model external walls in terms of geometry, composition, quantities, and extract information about them brings several benefits to the construction industry.

1.2 Model View Definitions

The mvdXML format [5] is an open-standard developed by buildingSMART, whose function is to define a subset of an IFC model. This subset is called a Model View Definition (MVD) and represents a certain part of the IFC data available in the model. The objective of an MVD is to ensure the delivery of the right and valid IFC data in the exchanges of the project actors, throughout the digital chain.

In order to do so, the MVD standard uses Concepts objects, which are the implementation of Concept Templates. Concept Templates represent a general notion such as the relationship between an element and associated properties, that can be applied to one or many entities in the model. This notion can be represented as a tree structure. For example, the notion of externality for an IfcWall element can be modelled as shown in Fig. 1.

The nodes of the Concept Template tree are Rule or Constraint objects. The AttributeRule nodes impose the name of attributes while the EntityRule nodes define the type that should be used for a value. Moreover, the nodes can be constrained by adding Constraint nodes. Those rules can also be expressed as logical expression, as detailed in [6, 7].

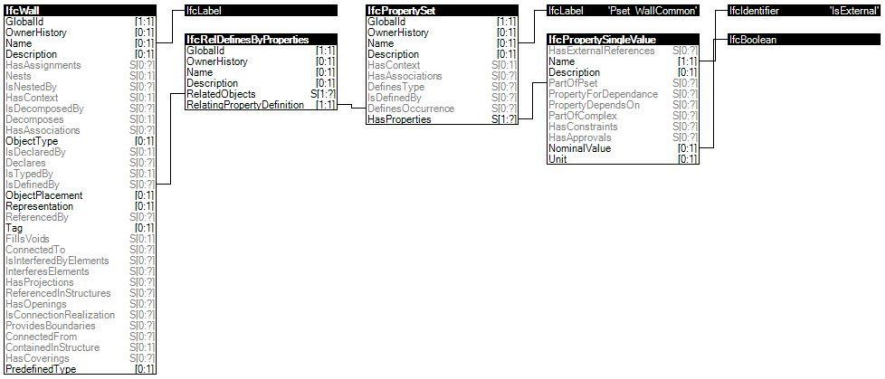

Fig. 1. The externality of IfcWall concept edited in the IfcDoc tool [8]
Overall an mvdXML file is a hierarchical structure forming a tree graph, with a root node mvdxml:root, and two main branches: the mvd:Templates, and the mvd:Views. In the templates branch, the Concept Templates are defined. In the Views branch, there can be one or several Concept Root nodes respectively representing an IFC entity towards which the concepts defined in the Concept Template nodes are applied. Moreover, Constraint nodes can be applied on the Rules nodes to impose a value of entity or attribute.

2 Literature review

2.1 Checking and validation

The diversity of disciplines and stakeholders involved in a construction project leads to an important number of rules to comply with [7, 9]. Indeed, checking whether a building project respects regulations require a set of numerous, precise, and reliable pieces of information about specific activities. BIM workflow allows to store, access, modify and share an unlimited amount of data about a construction project. The data enables to derive knowledge about the project and use it for rule checking, among other use cases [10]. However, the time saved will be only transferred from on operator to another if attributes values need to be analytically derived and then fulfilled to the model. The challenge is to write algorithms that can retrieve the implied knowledge of a building model by limiting the time-consuming tasks of data input and preprocessing [11]. To fulfill this objective, an assurance of the reliability of data is necessary. It depends on a strict formalization of the structure of the IFC information as well as compliance with minimal requirements on the model, such as minimal space requirements or absence of geometric collision [12].

The IFC model needs to respect storing requirements, that include the needs of application programs, in order to be useful for other applications than the one it has been originally edited on. In addition, rules are imposed on the data by regulatory conventions. Those rules have been acknowledged in previous research [6, 7] and concern the data existence and cardinality, the data content, the data uniqueness, and conditional dependency of data. The data existence imposes the presence of certain entities in the model, or as attributes of other entities. The data content requires a certain type or a certain value for data. The data uniqueness necessitates that for example only one GlobalId or one Name attributes should exist for any instance of an IfcRoot subclass. Finally, since there could be several ways to store data, the conditional dependency can be used in some cases to check the coherence of linked information.

2.2 State of the art of Model View Definition uses

As stated in [7], an MVD is an IFC subschema used for checking data compliance with diverse requirements. While is it mostly used to check the storing requirements, there are differences in the way MVD are embedded in a whole checking system. Moreover,

the output of the validation process is often a viewer displaying in highlighting the geometric representation of non-compliant IFC instances.

In the state of the art section of [6], the different MVD methods are listed and categorized according to various parameters: IDM, Model subset extraction, Validation Rules, Implementation of the validation function, OpenStandard, Edit tool. Among the five methods listed, none of them implements a validation function. In this work, the authors check the data storing structure of the model as described earlier. They give detailed examples of MVDs structure as well as a logical definition of the rules. The elements checked are part of BIM regulations. The output consists in BCF files that report the non-compliant entities in a shareable way.

In [13], mvdXML files are used to check space requirements for the design of a hospital. This work illustrates how MVDs can be implemented at the project scope, for a construction project. For example, this study mentions how the MVDs are defined with the building experts, to ensure interoperability between software applications. In addition, an effort is made to involve the building end-users in the MVD definition process, by fulfilling their requirements through CSV spreadsheets. The validation output is rendered in a viewer.

In [7] the MVD presented is used to comply with the Precast Concrete Industry (PCI) requirements, which shows the reusability of the method for a specific types of building elements, as in [6]. This research also provides a logical notation of the MVDs. Indeed, the set of rules defined by the PCI are registered into what the authors called a PCI MVD. The MVD is edited into the IfcDoc tool and a building project is checked against it. The results of the validation are shown within the IfcDoc tool.

More recently, Eastman [10] used a BACnet MVD that offers promising perspectives for data communication between IFC files and Building Automation System (BAS). This work allows the integration of a BAS in the design process, until now excluded from BIM workflow, and enables its operation by creating new IFC modeling types. A prototype is tested, which shows how BAS object families can be modeled in the authoring tool Revit. The BACnet MVD is used to export these Revit objects content to populate IFC files and import the data in a web-based application, considered as more accessible than the IfcDoc tool. This study demonstrates how an MVD formalization can fix the interoperability problem on use cases that had not been tackled until now and thus expand BIM benefits.

2.3 Conclusion

Regarding checking processes, the distinction between data and geometric checking is well marked but there are few works which attempt to fix semantic-geometric coherence of models [14, 15]. The identification of this issue is of great importance to accurately deliver high quality models, and it is where BIM offers real opportunities, when it enables to detect implicit knowledge as a domain expert would [16].

In cited research [13], MVDs are used as validation tools for data structure compliance with Exchange Requirements. Generally, studies agree on the benefits of mvdXML standards, attributed to metrics such as simplicity of edition and reusability

[6]. Current validation processes using MVDs consists in the edition of an mvdXML file, use a program to extract data from the IFC file, and eventually execute the mvdXML file on the IFC file. However, while there is a uniform way to define the requirements, the data extraction depends on a specific programming language, and set of tools. While it shows the strength of open standards, which is to read and edit data from any tools it also compromises reusability of such use cases.
In terms of the limitation of MVDs for data validation, [7] acknowledges that there is neither an easy nor convenient way for domain users to carry out the validation process by themselves. The IFC schema and mvdXML data model represent a barrier towards the adoption of this validation method. It is indeed necessary to provide user-friendly tools, as IfcDoc, to reduce computer science proficiency requirements from of the end-users.

Most of the exchange in a building project are not linear, and there is not always an order to follow in the validation. Efforts are concentrated on designing a system of exchange which can be hard to implement in a construction project. There are few mentions of a reactive approach instead, a tool or method that would allow any user in the chain to make the extraction and validation for its own use case. Generally, research assumes models have been validated throughout the phases of a project. However, these assumptions are valid when the collaboration is effective and BIM proficiency is advanced among the construction actors. In most situations, the model is wrong and there are few ways that give unfamiliar users to take advantage of the model it receives [7].

3 Contribution

3.1 Approach overview

We propose a set of open-source tools that use MVDs to extract, filter, and eventually export data from an IFC model coupled with corrective geometrical algorithms, assuring coherence between geometric and semantic information in the model. Our approach enables a wide variety of use cases such as quality assurance and code checking thanks to powerful extractive features. It relies on a formal and shareable definition of IFC data extraction, through mvdXML Concept Templates, and offers the possibility to export the data in an easily readable format such as a spreadsheet.

The method is based on the definition of one or multiple queries on the IFC file through an easily generable but formal and shareable specification in the form of MVDs. The gap existing between the information needed and the data formulation relies on knowledge about buildings and IFC format. For example, if the total surface of the façade is to be known, building practitioners must extract the surface quantity of all the external walls, assuming there are no curtain walls, and sum them up. Once that formulation is stated and transformed in an MVD, data can be extracted from the IFC file for domain specific activities.

As a drawback of the freedom offered by a neutral format, the data is not stored in uniformly in all IFC files [6] and makes it necessary to explore the model first in order to find how the hierarchical layout is articulated. For instance, for two different building

projects, the information about the wall area is stored in an IfcQuantitySet entity type whereas in the Duplex file it is stored in an IfcPropertySet entity type. In our approach, it obliges the user to build a new Concept Template per feature value. Therefore, we believe in a potential learning effect from this framework, enabled through the habit of editing trees of MVD to extract data.

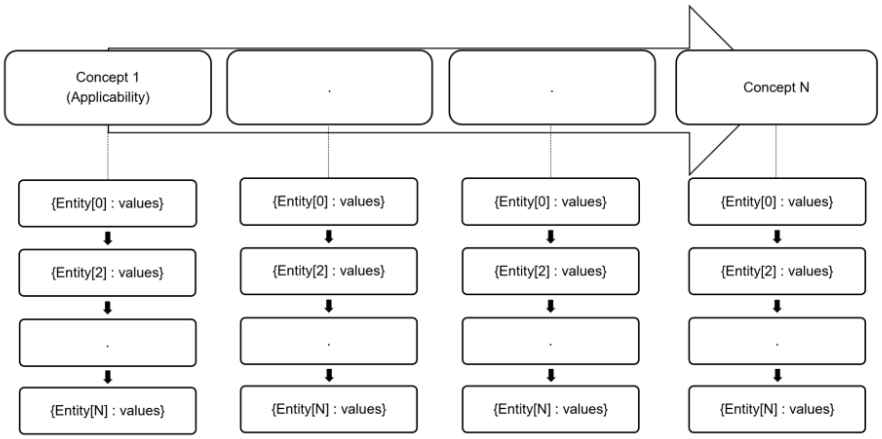

Fig. 2. The framework used for our approach. IFC values are collected throughout Concepts. The process shows the iteration over the Concepts, and the retaining of values mapped to entities. Those concepts can filter the entities as an mvdXML Applicability. For example, acts as a filter and only IfcWall instances pertaining to external walls will be extracted, also for each of the remaining concepts.

As shown in Fig. 2, the program developed through this research extracts data from Concept Templates units applied in series on IFC entities, whose type is determined through a Concept Root. Two functionalities exist in the program. First, as the entities pass the Concept Template units, the data required is collected. In addition, some Concept Templates can be made restrictive by constraining some of the values and thus excluding entities that do not comply with those constraints. Therefore, the data structure validation as in [6, 7] is also embedded in our extraction process, since data that do not comply will not be extracted.

3.2 Description of the extraction and validation algorithm for quantities of external walls

The extractive functionality of our program relies on a recursive function that extracts attribute values from IFC entity instances while traversing the MVD Concept Templates. First, the program maps an mvdXML file to a high-level programming

language to access its underlying structure. The mapping enables to store the Concept Template nodes applied on the ConceptRoot we want to extract data from in a variable to iterate over them. The program extracts the entities referenced by a ConceptRoot. Then, for each iteration over the Concept Template nodes, a traverse function is called, taking as input the entities and an mvdXML node, root of the considered MVD Concept Template tree, and will return a collection of EntityRule nodes mapping to literal value in IFC or literal values. Therefore, for each iteration, the results of the function correspond to one column of the spreadsheet export.

As presented in Fig. 3, the function recursively traverses the Concept tree while storing the data required by the AttributeRule, EntityRule, and Constraint mvdXML objects along the nodes. The behavior of the function differs based on the nature of its MVD node argument. Usually, for the initial call of the function all the IFC entities are passed with the root node of the Concept tree, an AttributeRule. The base case of the recursion occurs when the mvdXML node considered by the function is a leaf or when the child node of an EntityRule node is a Constraint. When the MVD node is an EntityRule the subset of currently supplied instances is taken that matches the IFC entity supplied and the recursive traversal continues. When the MVD node is a Constraint or leaf node the currently supplied literal values are passed to the higher-level invocation in the call stack. When the MVD node is an EntityRule there can be multiple AttributeRules meaning that at this point the results obtained from the lower level invocation on the call stack need to be combined. When the MVD node is an AttributeRule the different IFC instances can result in different attribute valuations so the size of the result set grows. For more details, we refer the reader to the open source repository containing the tools at https://github.com/opensourceBIM/python-mvdxml.

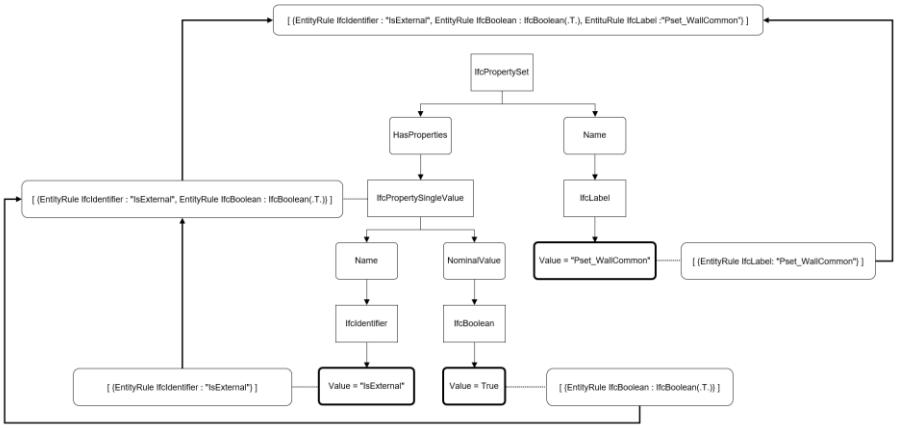

Fig. 3. Example of the extractive function applied on an IfcPropertySet EntityRule node. EntityRule, AttributeRule, Constraint are respectively represented by rectangular, slightly rounded edges, and bolded edges boxes. Output values are attached to the leaves by dotted lines and bolded arrows show the output process.

3.3 Limits of BIM and proposition of a corrective approach

MVDs are efficient tools to extract data from specific entities but reliability of information is necessary to bring a real value to the construction industry. Indeed, as stated previously, many models delivered in the professional sector lack coherence and comport errors. For example, there can be errors on the quantities directly calculated in the attribute values of the entities. Moreover, non-trivial information must also be verified, as the externality of elements such as the walls. The main use example of our work aims to show that it is possible to use easily edited MVD to export data to a well-known format, to gather the necessary quantity information necessary to carry out the cost estimation of façade insulation or run a thermal analysis. Nevertheless, this simple use case in theory is unfeasible without the assurance that the IFC data exported is reliable.

The models shared in practice are not reliable when received by a project actor. The exchange that appears throughout the digital chain can be a cause because each export uses its own export module and the accumulated errors result in a flawed model. Moreover, there can be modeling errors, such as not defining the spaces correctly. Finally, one of the most important sources of errors is the absence of systematic checking on project. And this is more globally due to an absence of BIM execution plan at the project scope. The use cases not defined before the beginning of the design phase results in uncontrolled exchanges.

Hopefully, primary -or low-level errors- can be fixed thanks to geometrical algorithms. Open source tools such as IfcOpenShell [17] enable access to the geometry description of IFC entities. Moreover, it also provides functionalities to access the schema at runtime, to delete the wrong entities values and their relationships, to eventually construct new entity instances and write them onto a newly corrected file. Almost all the geometric and data related can be fixed using such methods, but the difficulty is that there is no finite set of errors to check. It is therefore hard to fully automate data fixing for every case, especially when all the error cases are unknown. However, it can be done for frequently used data such as quantities or elements externality. The adoption of BIM by end-users relies on the confidence they have in the tool and their capacity to use it to fulfill their needs. It is better to be able to have basic information reliable than a lot of derived information based on false primary data.

As discussed in the Introduction section, the IFC schema offers for a wide variety of geometry definitions that are potentially complicated to evaluate. Existing software can be used to transform the implicit engineering constructs in IFC into an explicit and universal description called a Boundary Representation (BRep). It is not an approximation (as would be the case for a triangle mesh for example) but has full geometric fidelity for arbitrarily curved surfaces. A BRep is a hierarchical structure of topological items: Solid, Shell, Face, Loop, Edge, Vertex. A Face for example contains one or more Loops that bound an area over a surface. A Loop contains a pairwise connected sequence of Edges. Three of the topological constructs have associated geometry. A Face has an underlying Surface, an Edge is associated to a Curve and Vertex is located at a Point. In this research IfcOpenShell is used to convert the IFC geometry into the BRep format of Open CASCADE [18], the latter implements a wide variety of modeling and

analysis functionality, such as Boolean operations and the calculation of area and volume.

We consider an external wall as a wall that is not separating two internal spaces. For the simplicity of our example, we did not add additional constraints on the definition of externality for a wall. Checking the externality of a wall demonstrates the complexity of assigning a degree of trust to an IFC model and reinforces the need for the systematic verification of such cases. The IfcWall and IfcSpace representations enable to store the faces of those objects. Thus, we spot the internal walls as the one which have two of their biggest vertical faces intersecting shell representations of spaces. If a wall shares two common faces with a space, then it is not external.

name	surface	external
Basic Wall:Party Wall - CMU Residential Unit Dimising Wall:139234	12	TRUE
Basic Wall:Party Wall - CMU Residential Unit Dimising Wall:139347	12	TRUE
Basic Wall:Party Wall - CMU Residential Unit Dimising Wall:139374	0	TRUE
Basic Wall:Foundation - Concrete (417mm):140479	11	TRUE
Basic Wall:Foundation - Concrete (417mm):140520	22	TRUE
Basic Wall:Foundation - Concrete (417mm):140554	10	TRUE
Basic Wall:Foundation - Concrete (417mm):140602	21	TRUE
Basic Wall:Foundation - Concrete (435mm):140913	5	TRUE
Basic Wall:Foundation - Concrete (435mm):140987	5	TRUE
Basic Wall:Foundation - Concrete (435mm):141018	2	TRUE
Basic Wall:Party Wall - CMU Residential Unit Dimising Wall:143239	38	TRUE
Basic Wall:Exterior - Brick on Block:138157	50	TRUE
Basic Wall:Exterior - Brick on Block:138310	48	TRUE
Basic Wall:Exterior - Brick on Block:143410	17	TRUE
Basic Wall:Exterior - Brick on Block:143534	17	TRUE
Basic Wall:Exterior - Brick on Block:138062	14	TRUE
Basic Wall:Exterior - Brick on Block:138237	12	TRUE
Basic Wall:Exterior - Brick on Block:143478	41	TRUE
Basic Wall:Exterior - Brick on Block:143590	41	TRUE
Basic Wall:Exterior - Brick on Block:184944	5	TRUE
Basic Wall:Exterior - Brick on Block:185014	11	TRUE
Basic Wall:Exterior - Brick on Block:185064	5	TRUE
Basic Wall:Exterior - Brick on Block:185101	10	TRUE
	503	

Fig. 4. Illustration of the spreadsheet output resulting from our program which has successfully exported the Name (name column) and the Area (surface column) values of IfcWall instances whose IsExternal attribute value is set to true. The summation of the area demonstrates the simplicity of the use case.

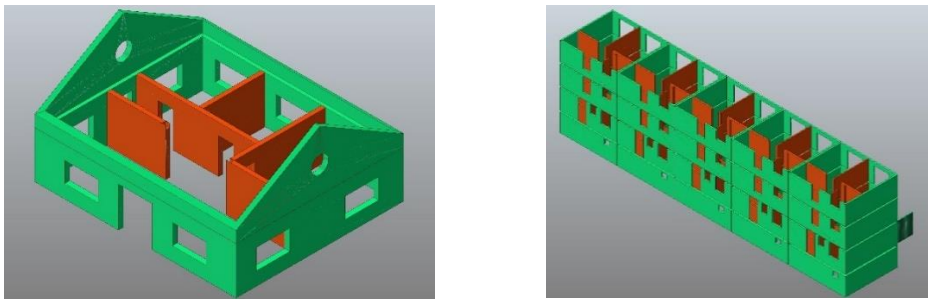

Fig. 5 . Visualization of the filtered entities on Haus Elite and Smiley-West files. Source: [19]

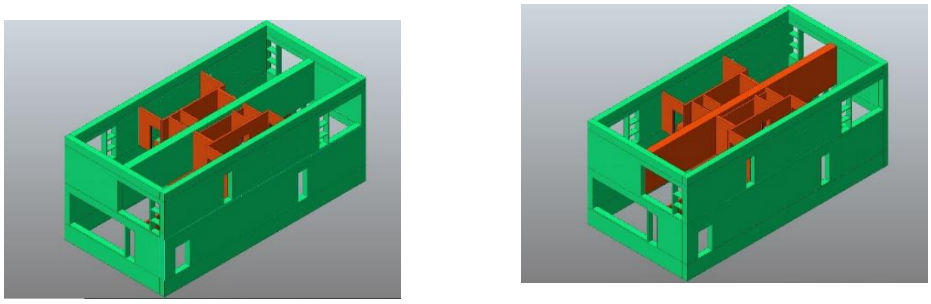

Fig. 6 . Visualization of the filtered entities (the internal walls) for the Duplex file before and after applying the geometrical fixing. The wall on the first floor was incorrectly classified as external before the fixing.

4 Discussion

4.1 The importance of data extraction

The ability to read IFC values is a fundamental part of checking a model or a project [9]. Indeed, the diversity of use cases and software applications has not delivered a unique way to validate data yet. We state that any validation operation requires a data extraction first. Before there could be any uniform way to validate data, extracting data could be formalized in a standard way.

The use of a formal and shareable extraction through mvdXML files encourages BIM adoption as different methods for querying make it a complex task for end-users. In several works, a different approach has been used to query the information of IFC files in order to check the rule compliance, whereas extraction is at the heart of BIM. However, MVD was often used for software use cases, which was essentially checking the data structure, uniqueness, or existence. With our approach, mvdXML files can be used to directly extract information about the model for end-users in an easily accessible format. Therefore, we think of our approach as a method reducing the gap between

end-users programming proficiency needs and the ability to exploit IFC data, while getting familiar with the mvdXML open-standard.

4.2 Results discussion

Extracting the data and executing a validation regarding an MVD offers a real advantage for a whole range of users. Indeed, our approach can also be used by more experienced users who wish to reduce complexity of extraction and validation. We therefore hope to provide a robust support for MVD validation of IFC data.

The most common uses of MVDs can be extended to data extraction, for example in an easily readable way such as a spreadsheet for end-users. Also, MVDs can be constrained in a way so that IFC entities not complying with mvdXML Rules will not be extracted, thus marked as not conform. Eventually, filtered entities can be rendered in a viewer application.

Results are successfully extracted from the MVD and displayed in Fig.5. and Fig.6. The results from Fig. 6 demonstrate that some external walls been have correctly fulfilled by the model's authors. However, there are still errors that cannot be totally spotted by our algorithm. For example, on Fig.6, the IsExternal value of a wall was not correctly classified and could lead to estimation errors if repeated on bigger projects. The geometric algorithm used to identify external walls relies on a proper definition of entities and a correct modeling of spaces. Without these conditions, it misclassifies the walls regarding to their externality. For example, Fig.7 shows the walls identified as external by the algorithm. Although the result may seem correct, there are still errors, especially small elements which are IfcWall instances, but which does not really represent a wall. Regarding this topic, Krijnen [16] implemented an unsupervised machine learning algorithm to carry out anomaly detection on the Duplex model, resulting in the discovery of misclassified elements.

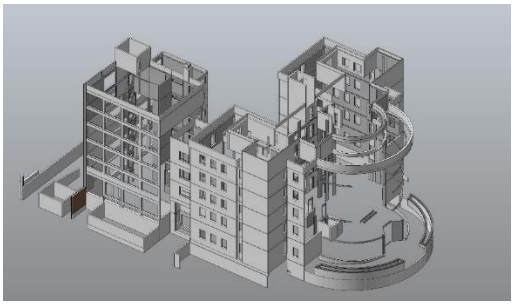

Fig. 7 . Visualization of external walls on a model from the construction industry after applying the fixing algorithm. The model was delivered without any IsExternal classification, since all the walls were marked as external.

4.3 Validation levels and follow-up works

IFC models must be validated, particularly regarding their semantic-geometric coherence, before using them to check project compliance with building codes. There is still a vast majority of models containing lacking, unverified, wrong, or misleading information. A model that contains errors, or whose data quality has not been checked appropriately, should not represent a trusty basis for real-life checking use cases such as structural analysis or cost estimation [14].

In the same way [9] proposes a hierarchy of rules, we believe in the necessity of the construction of a hierarchical level of trust for IFC models. This level of trust should start by checking that IFC instances representation correspond to their nomination, such as what has been initiated by Krijnen [16]. Then, the model should be observed and corrected. Our approach wishes to accomplish those use cases. Nevertheless, the identification of geometric-semantic potential issues should be made, as we did for instance for walls externality and elements quantities. Finally, the structuration of the data in the model, and the real-life checking will be properly be carried out.

5 Conclusion

This research seeks to highlight the opportunities offered by mvdXML files to extract data destinated to domain specific use cases from IFC files, as well as to show IFC values can be corrected by processing the underlying description of their geometry. In the Introduction part, we demonstrated the role of IFC files and how we can use them to ease completion of use cases that would bring significant advances in the construction industry. The repository containing the set of tools used in our approach is available on https://github.com/opensourceBIM/python-mvdxml. We also showed MVD use to validate and filter IFC values and how it provides value to the digital transmission chain, improves models' quality, and offers promising perspectives through new use cases.

However, we noticed that while our results are encouraging, difficulties persist:

- In the industry, many of the models are made of errors and BIM processes not rooted yet into practice.
- MVDs rely upon a project-based organization responsible for the BIM workflow processes, such as data exchanges. Current approaches seek to forecast all the use cases rather than to give every project member the power to extract and check data independently.
- The end-users often cannot use the IFC data in an independent way. They need a viewer application or an authoring tool, which is an intermediary between themselves and the IFC data, that can hide the full information extraction and validation possibilities offered by IFC and MVD standards.
- The data extraction and checking are usually made within software applications from a GUI, without the possibility to reuse rules from one application to another.

Assessing these difficulties and aware of their impacts, we developed an approach seeking to:

- Show how simple a query can be translated into an mvdXML Concept tree
- Give the ability for end-users to extract data in a familiar format to gain independence
- Give the ability for most experienced user to build upon our work to develop more complex parsing programs and elaborated uses of MVDs
- Show a non-trivial to detect example of errors in the building model
- Prove that it can be corrected with geometrical algorithms
- Provide a flexible way to work with IFC data and process it digitally rather than depending on viewers or authoring tools
- Acknowledge the unknown regarding the IFC values to correct and how to check them

Regarding the last point, we are open to collaborate on verifying the basic data of IFC files, which can be wrongly identified, such as the space definition or the IFC entity types, in order to ensure a high level of confidence for the most used IFC entity types .

References

1. Solihin, W., Eastman, C., Lee, Y.-C.: Toward robust and quantifiable automated IFC quality validation. Adv. Eng. Inform. 29, 739–756 (2015). https://doi.org/10.1016/j.aei.2015.07.006
2. Wang, W., Zmeureanu, R., Rivard, H.: Applying multi-objective genetic algorithms in green building design optimization. Build. Environ. 40, 1512–1525 (2005). https://doi.org/10.1016/j.buildenv.2004.11.017
3. Zemella, G., Faraguna, A.: Evolutionary Optimisation of Façade Design: A New Approach for the Design of Building Envelopes. Springer-Verlag, London (2014)
4. Evins, R.: A review of computational optimisation methods applied to sustainable building design. Renew. Sustain. Energy Rev. 22, 230–245 (2013). https://doi.org/10.1016/j.rser.2013.02.004
5. Chipman, T., Liebich, T., Weise, M.: Specification of a standardized format to define and exchange Model View Definitions with Exchange Requirements and Validation Rules.
6. Zhang, C., Beetz, J., Weise, M.: Model view checking: automated validation for IFC building models. In: Mahdavi, A., Martens, B., and Scherer, R. (eds.) eWork and eBusiness in Architecture, Engineering and Construction. pp. 123–128. CRC Press (2014)
7. Lee, Y.-C., Eastman, C.M., Solihin, W.: Logic for ensuring the data exchange integrity of building information models. Autom. Constr. 85, 249–262 (2018). https://doi.org/10.1016/j.autcon.2017.08.010
8. IfcDoc, https://technical.buildingsmart.org/resources/ifcdoc/

9. Solihin, W., Eastman, C.: Classification of rules for automated BIM rule checking development. Autom. Constr. 53, 69–82 (2015). https://doi.org/10.1016/j.autcon.2015.03.003
10. Tang, S., Shelden, D.R., Eastman, C.M., Pishdad-Bozorgi, P., Gao, X.: BIM assisted Building Automation System information exchange using BACnet and IFC. Autom. Constr. 110, 103049 (2020). https://doi.org/10.1016/j.autcon.2019.103049
11. Gu, N., London, K.: Understanding and facilitating BIM adoption in the AEC industry. Autom. Constr. 19, 988–999 (2010). https://doi.org/10.1016/j.autcon.2010.09.002
12. Azhar, S.: Building Information Modeling (BIM): Trends, Benefits, Risks, and Challenges for the AEC Industry. Leadersh. Manag. Eng. 11, 241–252 (2011). https://doi.org/10.1061/(ASCE)LM.1943-5630.0000127
13. Weise, M., Nisbet, N., Liebich, T., Benghi, C.: IFC model checking based on mvdXML 1. 9 (2016)
14. Häußler, M., Borrmann, A.: Model-based quality assurance in railway infrastructure planning. Autom. Constr. 109, 102971 (2020). https://doi.org/10.1016/j.autcon.2019.102971
15. Daum, S., Borrmann, A.: Checking spatio-semantic consistency of building information models by means of a query language. Presented at the (2013)
16. Krijnen, T., Tamke, M.: Assessing Implicit Knowledge in BIM Models with Machine Learning. In: Thomsen, M.R., Tamke, M., Gengnagel, C., Faircloth, B., and Scheurer, F. (eds.) Modelling Behaviour. pp. 397–406. Springer International Publishing, Cham (2015)
17. IfcOpenShell, http://ifcopenshell.org/
18. OPEN CASCADE, https://www.opencascade.com/
19. Open IFC Model Repository, http://openifcmodel.cs.auckland.ac.nz/