Final published version is found in the ASCE Library here: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000909
Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

A Data-Driven Reverse Engineering Algorithm Development (D-READ) Method for Developing Interoperable Quantity Takeoff Algorithms Using IFC-Based BIM

Temitope Akanbi[1]; Jiansong Zhang, Ph.D., A.M.ASCE[2]
and Yong-Cheol Lee, Ph.D. A.M.ASCE [3]

Abstract

One main gap in the automation of construction quantity takeoff (QTO) is the lack of a systematic method to address building information models (BIMs) created from different BIM authoring tools/workflows. Even the industry foundation classes (IFC), one of the ISO standard data schema, has been used in a variety of ways, some of which could be proprietary. To address this gap, the authors proposed a new Data-Driven Reverse Engineering Algorithm Development (D-READ) method for developing QTO algorithms based on IFC geometric analysis. The proposed method enables the development of QTO algorithms for IFC-based BIMs resulted from different BIM authoring tools/workflows and therefore enhances robustness of BIM-based QTO. It takes a novel bottom-up approach in QTO algorithm development comparing to the traditional top-down approach. A model view definition (MVD) model for IFC model checking was developed and incorporated with the QTO algorithms. The proposed method was tested on nine different BIM instance models from different sources. A comparison with state-of-the-art commercial

---

[1] Graduate Student, School of Construction Management Technology, Purdue University, West Lafayette, IN 47907. E-mail: takanbi@purdue.edu

[2] Assistant Professor, School of Construction Management Technology, Purdue University, West Lafayette, IN 47907 (corresponding author). E-mail:zhan3062@purdue.edu

[3] Assistant Professor, Department of Construction Management, Louisiana State University, Baton Rouge, LA 70803; E-mail: yclee@lsu.edu

Final published version is found in the ASCE Library here: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000909
Akanbi, T., Zhang, J., and Lee, Y.C. (2020). “Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM.” J. Comput. in Civ. Eng., 34(5), 04020036.

software showed consistent QTO results whereas the proposed D-READ method resulted in QTO algorithms that were more robust with regard to the different BIM authoring tools/workflows used.

Keywords: Quantity Takeoff, Industry Foundation Classes, Automation, BIM Interoperability, Model View Definition.

1. Introduction

Cost estimation is critical to the success of a construction project (Yu et al. 2006; Choi et al. 2015). Manual cost estimation is a tedious, time-consuming, and cumbersome task that usually involves human errors (Samphaongoen 2009). Studies showed that building information modeling (BIM) tools can provide benefits to owners and construction professionals through cost estimation automation. However, a review on BIM literature between 2005 and 2015 showed that only a few BIM studies focused on cost estimation (Santos et al. 2017). Present BIM research heavily focused on other tasks such as 3D coordination, clash detection (Kreider et al. 2010; Franz and Messner 2017), safety checking (Zhang et al. 2015) and facility management (Liu and Issa, 2013, 2014, 2015). While futuristic BIM studies already look into cyber-security in cloud computing (Mutis and Paramashivam 2019) and human BIM robot interactions (Mutis et al. 2019), much research is still needed to support eminent BIM development for its practical applications to meet the increasing demands of the architecture, engineering, and construction (AEC) industry (McGraw-Hill Construction 2014). Automated quantity takeoff (QTO) is one of the most useful such development (Monteiro and Martins 2013; Franco et al. 2015; Plebankiewicz et al. 2015). Currently, several methods, techniques, and software programs are available for QTO purposes. However, most commercial software programs such as D-profiler, Autodesk Revit, Assemble and Navisworks, Sigma Estimates, and Vico Estimator use their proprietary data formats which cause errors and missing data when exchanging model with other software or are not able to exchange data with other software at all (Lee et al. 2014). Such lack of interoperability hinders the adoption of BIM in QTO automation to realize its benefits. The use of a standardized format such as industry foundation classes (IFC) may solve this interoperability (successful data exchanges) problem and enable wide adoption of BIM in QTO automation (Choi et al. 2015; Zhang 2018).

There are different QTO solutions geared towards solving specific issues in the construction industry. In the 2D realm, there are tools that enable construction professionals to digitally perform QTO using 2D drawings such as eTakeoff, On-Screen Takeoff, PlansSwift, etc. In the use of 3D BIM, which is the focus of this research, there are advanced tools that enable construction professionals to perform model-based QTO such as Sigma Estimates and Navisworks. However, despite the advent of these 3D BIM-based QTO solutions, there are major barriers in their wide adoption due to the interoperability problem. Such QTO solutions require the importation of a building design in BIM, but they are not necessarily compatible with all BIM authoring tools/workflows.

To solve this interoperability problem, an IFC-based approach is widely accepted as the most promising direction. There have been many researches investigating information extraction from IFC-based BIM for various purposes (Zhang and El-Gohary 2015a, Ding et al. 2017, Ramaji and Memari 2018a), and few among them focused on QTO from IFC-based BIM (Choi et al. 2015, Ma et al. 2013). However, there is a lack of a systematic method to address QTO from building information models (BIMs) created from different BIM authoring tools/workflows.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Such a capability is critical to enable a wider adoption of BIM-based QTOs. To address this gap, the authors propose a new Data-driven Reverse Engineering Algorithm Development (D-READ) method which is an iterative method to generate QTO algorithms that can cover a variety of (and eventually all) types of BIM representations in IFC.

2. Background

2.1. BIM Interoperability and Industry Foundation Classes (IFC)

Although BIM was intended to be interoperable since its introduction, a seamless BIM interoperability is far from reality (Nawari 2012; Cheung et al. 2012; Santos et al. 2017; Zhang 2018; Ramaji and Memari 2018a). With the growing predominance of BIM in the construction industry, the most common synergy across BIM applications is still a one-to-one relationship (Lai et al. 2018). However, interoperability based on such one-to-one relationship is inefficient because it would require the development of C^2_n conversion algorithms for interoperability between n BIM software. In comparison, interoperability based on a many-to-one relationship would be much more efficient. The industry foundation classes (IFC) standard is widely accepted as the most promising potential solution for BIM interoperability enabled by the many-to-one relationship (Wu and Zhang 2019) which helps form a "closed-world response to an open-world problem" (Costin and Eastman 2019). It is an ISO registered data standard for building and construction industry data (ISO 16739) and provides an open and neutral platform for information exchange within the AEC industry in a standardized way. Much research efforts have developed IFC-based approaches in solving interoperability issues in the construction domain. For example, Hernandez et al. (2018) addressed the lack of interoperability between the multiple equipment deployed on site to perform self-inspection of buildings through developing a framework that

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

utilizes an IFC-based approach to merge data from different resources for use in such building inspection applications. Ding et al. (2017) developed an IFC-based inspection process to fuse inspection data represented in different formats and stored in different locations to enable real-time quality monitoring and control. Zhang and El-Gohary (2013, 2015b) developed an information extraction and transformation method that automatically extracts information from textual building regulations and transforms the information into computable logic rules to check design information from IFC-based BIMs, to enable interoperability between BIM and textual information systems. Ramaji and Memari (2018a) developed an Interpreted Information Exchange (IIE) mechanism for transformation of IFC-based BIMs in the Coordination View to their equivalent structural models in IFC Structural Analysis View.

Golabchi and Kamat (2013) stated that although interoperability among various BIM software applications can be achieved using IFC data exchange, complete interoperability cannot be achieved until the BIM authoring tools become IFC-certified. Moreover, even between BIM authoring tools that are IFC-certified, a complete interoperability is still not guaranteed. For example, Choi and Kim (2011) conducted a study to test interoperability between IFC-certified BIM-based environmental analysis software by exporting and importing IFC files. The results found that although the software supports IFC, the data exchange between the tools wasn't seamless. One main reason for such incapability to achieve seamless interoperability through IFC-based data exchange is the lack of conformity between the ways IFC schemas are adopted by individual BIM tools (Steel and Drogemuller 2009; Cheung et al. 2012; Zhang 2018; Wu and Zhang 2019). The IFC files generated from different platforms vary significantly (Sun et al. 2015). For example, Lee et al. (2011) conducted an experiment by comparing the similarities and

Final published version is found in the ASCE Library here: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000909
Akanbi, T., Zhang, J., and Lee, Y.C. (2020). “Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM.” J. Comput. in Civ. Eng., 34(5), 04020036.

differences in the IFC data for the same model but exported from two different BIM applications (Revit and ArchiCAD), which found only about 54% of common IFC entities between the two exports.

2.2. IFC-based Quantity Takeoff

Few researchers explored IFC-based QTO. For example, Drogemuller (2003, 2005) introduced an automatic estimator that takes IFC-based BIM as input and automatically generates a bill of quantities for “reinforced concrete, post tensioning, formwork, masonry, and steel work”. Ma et al. (2013) developed an IFC-based semi-automatic cost estimation model that can take off quantities according to the Chinese standard GB50500 for bill of quantity of construction works. Choi et al. (2015) developed a statistical calculation method that extracts quantities from IFC-based architectural elements for material QTO. However, there is a lack of IFC-based QTO methodology that supports data created from different BIM authoring tools/workflows that may use IFC entities and attributes in different ways.

2.3. Model View Definitions

The quality of IFC models varies and therefore the accuracy of an IFC instance file exported from BIM authoring tools needs to be evaluated (Weise et al. 2009). The National BIM Standard (NBIMS) was established in an effort to eliminate the uncertainties of information exchange between the users of BIM information (Lee et al. 2016). The development of such information exchange frameworks introduced by buildingSMART entails the following two foundational components - the information delivery manual (IDM) and the model view definitions (MVDs). IDM, which is the aggregated specifications of BIM data exchange requirements defined for a specific discipline, plays a pivotal role to provide a baseline for developing MVD with the IFC schema. MVDs provide comprehensive specifications of the BIM data exchange translated from domain knowledge in the IDM into the IFC schema (Eastman et al. 2009; Lee et al. 2016). Over the last two decades, several MVDs have been developed to support and enhance interoperability (Ramaji and Memari 2018b). Despite the establishment of this standard, there are still gaps in data exchange processes because of data mapping errors of IDM and MVD, insufficient consensus of domain experts, and translation problems from/to native BIM models to/from IFC instance files (Lee et al. 2015; Lee et al. 2016). To evaluate whether BIM data fulfil data exchange requirements, an MVD-based checking should be adopted to validate the accuracy of the IFC file (Lee et al. 2019). An MVD consist of a sequence of specification units referred to as 'concept,' which includes blueprint of IFC entities, their attributes, relationships, and properties (Venugopal et al. 2012). MVDs pinpoints portions of an IFC data structure supported within a particular model view (buildingSMART 2011). One of the main characteristics of an MVD is its reusability, allowing these concepts to be continuously applied in developing other specifications across several domains (Lee et al. 2018). An MVD allows a user to declare the necessary attribute/entity relationships for the specific use of the IFC file such as QTO.

3. Proposed Method

To address the research gap in IFC-based QTO that supports BIM data from different sources (i.e., BIM authoring tools/workflows), the authors propose a new data-driven method for developing automated QTO algorithms using IFC-based BIMs. This method can be utilized to develop QTO algorithms for any building component and the developed QTO algorithms can be applied to models created from any IFC-compatible BIM authoring tools/workflows. The authors named it Data-driven Reverse Engineering Algorithm Development (D-READ) method, which includes five main steps to reverse engineer the found representations of building components in an IFC model to develop algorithms for QTO purposes (Fig. 1). Step 1: Model development, this step identifies or establishes 3D models in different BIM authoring tools/workflows; Step 2: Model View Definition (MVD) development, this step creates an MVD for checking if an input IFC model contains the necessary information needed for QTO, e.g., geometric attributes. This step includes three sub-steps: scope identification, exchange requirements definition, and IFC entity mapping; Step 3: Data analysis, this step includes object identification and geometric representation analysis. Step 4: Algorithm development, this step reverse engineers quantity takeoff algorithms based on data analysis results; and Step 5: Algorithm implementation and testing, this step implements the developed QTO algorithms, tests the performance, and iteratively improves the algorithms through testing until a satisfying performance is achieved. The D-READ is not an algorithm nor a software per se but a method for developing interoperable QTO algorithms for BIM. The research question that is being addressed is whether such a data-driven, reverse engineering method could produce more robust QTO algorithms than what is available in the state of the art.

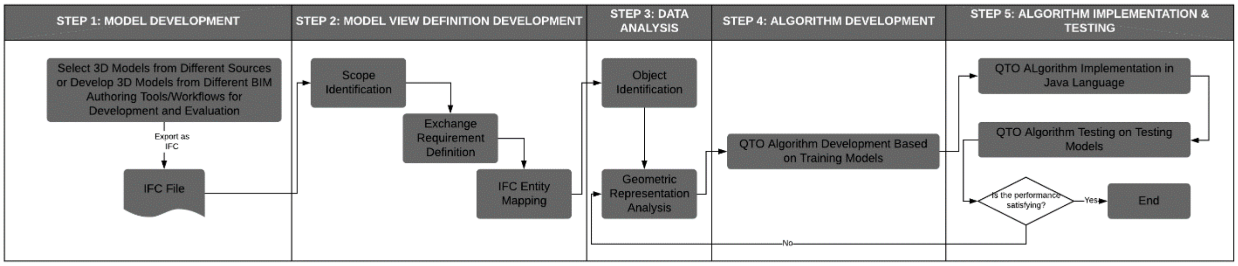


Fig. 1. Proposed Data-Driven Reverse Engineering Algorithm Development (D-READ) Method

An illustration of how developed QTO algorithms as a result of the D-READ method should be applied is shown in Fig. 2. There are two main processes, (1) MVD-based checking, and (2) applying the developed QTO algorithms. The MVD-based checking uses the MVD developed from Step 2 of the D-READ method to check an input IFC model and informs users to provide more input if the model does not contain all necessary information needed for QTO. If the input IFC model contains sufficient information, the developed QTO algorithms automatically extracts the quantities of building components from it. The developed algorithms achieve these by analyzing the model-specific geometric representations of building objects, which are based on arbitrary choices made in the proprietary BIM authoring tools/workflows, under the constraints set by IFC schemas.

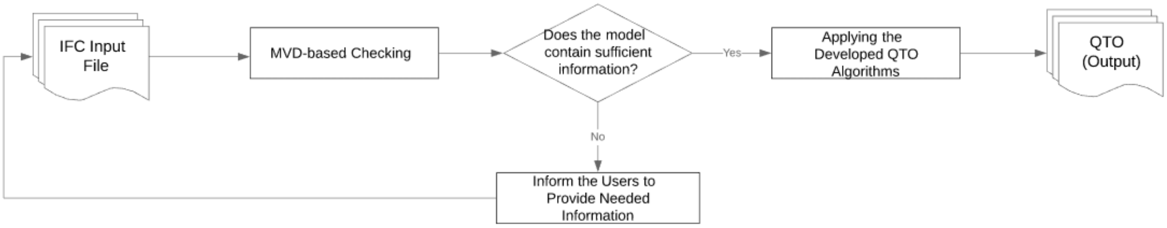


Fig. 2. An Illustration of the Application of Developed QTO Algorithms

The proposed D-READ method takes IFC models as input, which can be obtained from many different BIM authoring tools/workflows. According to buildingSMART (2019), eighty-three BIM software platforms are IFC-certified and therefore compatible with IFC. Different workflows can be built based on these BIM platforms together with other BIM/Non-BIM platforms. As an example, an original architectural design in ArchiCAD can be exported to IFC and imported into Autodesk Revit, however, manual modifications might be needed in Autodesk

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Revit. The details of the five steps of the D-READ method are described in the following subsections, respectively.

Step 1: Model Development

In this step, the model data to be used in QTO algorithm development and evaluation are selected or generated, including training data and testing data for algorithm development, and evaluation data for algorithm evaluation. The training data are used to develop the QTO algorithms. The testing data are used to test the developed QTO algorithms for potential improvements. The evaluation data are used to evaluate the robustness of the developed QTO algorithms. To cover different possible IFC entity/attribute usage patterns, the similar model data are created from different BIM authoring tools or workflows. For example, the same building design can be created using Autodesk Revit, Trimble SketchUp, GRAPHISOFT ArchiCAD, and other BIM authoring tools/workflows. Existing models can be used if their sources or creation workflows are known. The only constraint is that they should be able to convert to IFC data, either through direct exportation in a selected BIM authoring tool, or through proprietary or third-party conversion tools/workflows. For each source or authoring tool/workflow, there should be a training model, a testing model and an evaluation model.

Step 2: Model View Definition (MVD) Development

In the development of the MVD, there are the three following sub-steps: scope identification, exchange requirement definition, and IFC entity mapping. In an AEC project, there are different user groups (e.g. clients, architects) requiring information for different applications (e.g. thermal comfort analysis, cost estimation). In the scope identification sub-step, the user group/applications for which information is to be exchanged is identified. In the sub-step for

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

exchange requirement definition, the functional requirements for the information exchange are identified and organized into a set of MVD concepts. An MVD concept is defined based on a concept template and in reference to an IFC entity. A visual representation of entities and attributes involved in this MVD concept, as well as constraints and parameters set for selected attributes and entity instance types is created. Optional and mandatory entities for the data exchange are also defined according to IDM specifications. In the IFC entity mapping sub-step, the MVD concepts are mapped to IFC entities where the attributes and constraints are also mapped to the corresponding components according to the IFC schema.

Step 3: Data Analysis

In this step, there are two sub-steps: (1) object identification; and (2) geometric representation analysis. The two sub-steps are described in detail below:

Object Identification

The object identification step is necessary for deriving the needed information required in developing the QTO algorithms. As shown in Fig. 3, the operations of this step include preprocessing, object interpretation creation, and object interpretation validation. This is necessary to determine how objects are represented using the IFC schema, that is, what are the important attributes that differentiate each AEC object. First the IFC files are preprocessed – filtered and segmented so that only the entities and attributes related to the target object remain. Secondly, the object interpretation creation is performed, which is a determination of how objects can be represented therefore identified. Thirdly, the object interpretation validation is performed by verifying the representation interpretation with a collection of examples. For example, a wall is usually (but not always) represented using an *IfcWallStandardCase* instance in IFC, with the following attributes as per the IFC schema: *"GlobalId," "OwnerHistory," "Name," "ObjectType," "ObjectPlacmenet," "Representation,"* and *"Tag."* The IFC schema also showed the *Representation* attribute uses an *IfcProductDefinitionShape* entity which, in turn has an attribute called "*Representations*," which is a list of representations. However, the IFC schema does not specify how many representations there should be in the list, and what type each representation will be. Thus, not until training data is analyzed that it could be figured out where/how to process the geometric representation of this wall object. A simple piece of IFC data showing the use of two representations (i.e., one for "body" and one for "axis") in the representation list of a wall is shown in Fig. 4. Such arbitrary choice in the use of entities and attributes can occur throughout the IFC data, and this is what the object interpretation sub-step is designed to address.

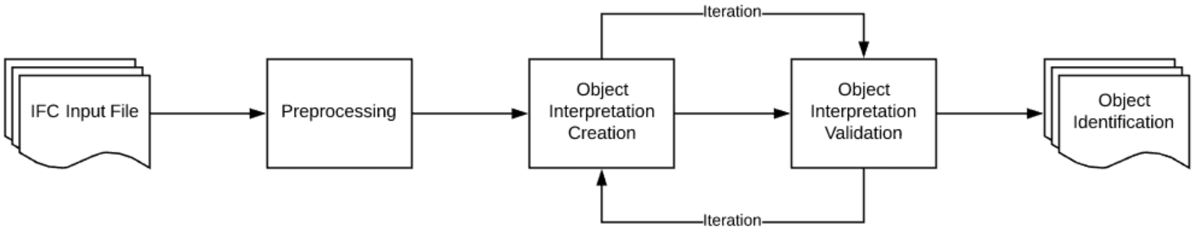


Fig. 3. Object Identification Processes

Geometric Representation Analysis

In this step, the IFC files are further analyzed. Specifically, the patterns in the use of IFC data structure and the attributes of the target component's geometric representations in the IFC data are analyzed (Fig. 4). The analysis result is used to create data tracing patterns for QTO purposes. To illustrate this process, the tracing patterns in retrieving the height (*WallDepth*), length (*XDim*), and width (*YDim*) of a rectangular wall will be explained below.

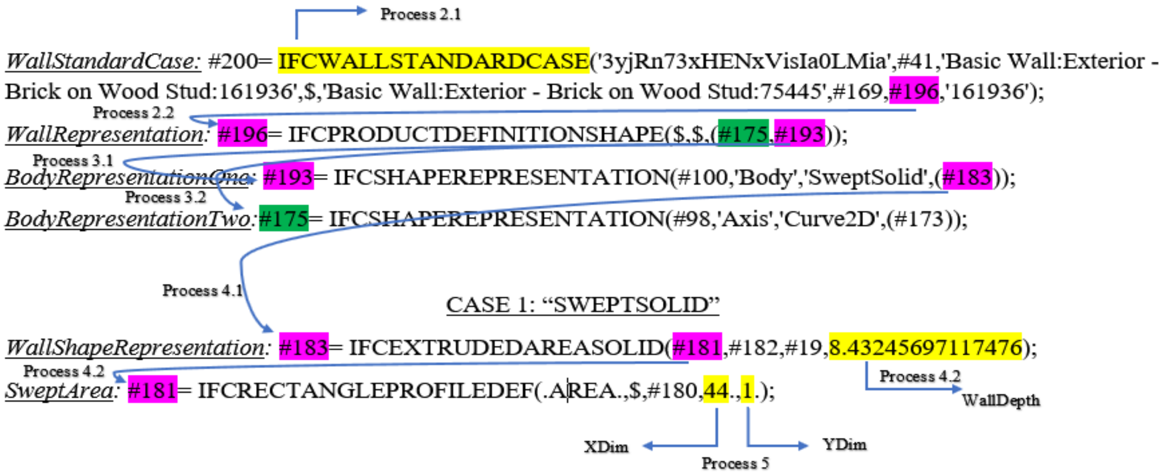
Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

As shown in Fig. 4, *Process 2.1* extracts the "IFCWALLSTANDARDCASE" entity into a variable named *WallStandardCase.* Process 2.2 extracts the seventh attribute of *WallStandardCase*, which is an "IFCPRODUCTDEFINITIONSHAPE," as *WallRepresentation*. Process 3.1 extracts the second element of the third attribute of *WallRepresentation*, which is an "IFCSHAPEREPRESENTATION," as *BodyRepresentationOne*. Process 3.2 extracts the first element of the third attribute of *WallRepresentation*, which is an "IFCSHAPEREPRESENTATION," as *BodyRepresentationTwo*. There are several representation types for shape representations. In "*CASE 1,*" the representation type is *"SweptSolid."*

CASE 1: "SweptSolid" BodyRepresentation

If *BodyRepresenationOne* is using the "SweptSolid" type of shape representation, *Process 4.1* extracts the first element of the fourth attribute of *BodyRepresentationOne,* which is an "IFCEXTRUDEDAREASOLID," as *WallShapeRepresentation*. *Process 4.2* further extracts the first and fourth attributes of *WallShapeRepresentation*, as *SweptArea* and *WallDepth* (i.e., the height of the wall), respectively. If the *SweptArea* uses "IFCRECTANGLEPROFILEDEF," *Process 5* extracts the fourth and fifth attribute of the entity as *XDim* (i.e., the length of the wall) and *YDim* (i.e., the width of the wall), respectively.

Final published version is found in the ASCE Library here: https://doi.org/10.1061/(ASCE)CP.1943-5487.0000909
Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.



```
WallStandardCase: #200= IFCWALLSTANDARDCASE('3yjRn73xHENxVisIa0LMia',#41,'Basic Wall:Exterior - Brick on Wood Stud:161936',$,'Basic Wall:Exterior - Brick on Wood Stud:75445',#169,#196,'161936');
WallRepresentation: #196= IFCPRODUCTDEFINITIONSHAPE($,$,(#175,#193));
BodyRepresentationOne: #193= IFCSHAPEREPRESENTATION(#100,'Body','SweptSolid',(#183));
BodyRepresentationTwo:#175= IFCSHAPEREPRESENTATION(#98,'Axis','Curve2D',(#173));

CASE 1: "SWEPTSOLID"

WallShapeRepresentation: #183= IFCEXTRUDEDAREASOLID(#181,#182,#19,8.43245697117476);
SweptArea: #181= IFCRECTANGLEPROFILEDEF(.AREA.,$,#180,44.,1.);
```

Fig. 4. Tracing Pattern of a "SweptSolid" Representation of a Rectangular Wall

Step 4: Algorithm Development

This step develops the QTO algorithms for taking off the needed linear, areal, and/or volumetric quantities of the analyzed building object. The algorithms follow the tracing patterns identified in Step 3 to extract the needed parameters and perform quantity computations using these parameters to obtain the needed quantities.

Step 5: Algorithm Implementation & Testing

The developed algorithms are implemented in a java program and tested on the testing data. In the development of the java program, java implementation methods are created to access the different building elements classes (e.g., walls, floors, stairs, roofs) to identify a building element and the corresponding QTO algorithm that needs to be activated. A Unified Modeling Language (UML) class diagram is used to help design the structure of the program. The UML diagram describes the system by showing the classes, the attributes, the operations and the interrelationships between the classes (Fig. 5). For example, "Room" and "Component" are two class elements of the system. A "Room" has multiple "Components", where a single

"Component" can only belong to one "Room." Therefore, the two classes have a one-to-many relationship. The "Component" class has several sub-classes: the "Floor," "Wall," "Stair," and "Roof" classes, which inherit all the attributes of the "Component" class and have additional attributes to satisfy the modeling of each type of building component, respectively. The "Wall" and "Stair" classes could either be "Interior" or "Exterior" while the "Roof" class could have several different types (e.g., flat roof, gable roof). The "Visitor" class is used to declare the visit operations for the "Component" classes. The "Visitor" class has a subclass "QTOVisitor" used for the QTO operations; Other visitor subclasses can be further added to extend the computational operations.

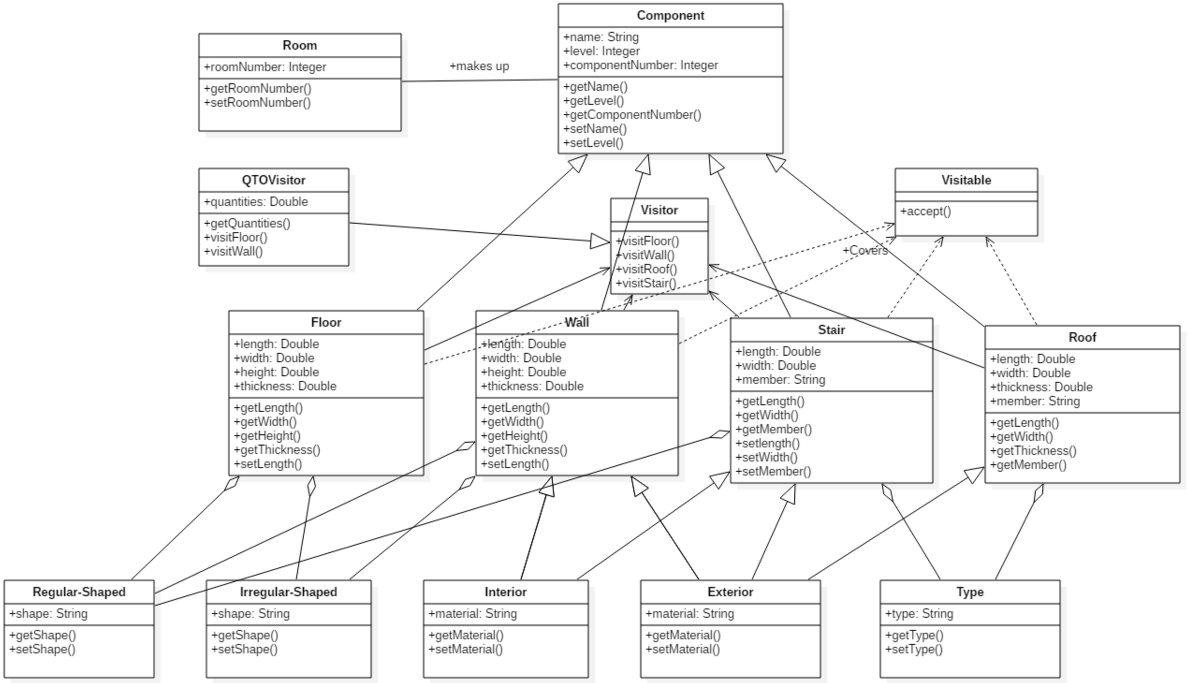


Fig. 5. UML Class Diagram for the Algorithm Development

Experiment

To test the effectiveness of the proposed D-READ method, the authors conducted an experiment with details described in the following subsections.

Experimental Setup

The system's setup composed of a 64-bit windows laptop with a 17" screen connected to two 17" display monitors via a VGA cable and an optic mouse. The laptop was operating on Windows 10 pro; the processor was an Intel ® core ™ i7 – 3720 QM CPU @ 2.60 GHz and the RAM was 16GB. The computing system's specifications – a system's memory (RAM) and storage affect the speed at which a system performs computational operations.

QTO Algorithm Development Using the D-READ Method

Step 1: Model Development

Nine building models (Models *A* – *I*) were used for training, testing and evaluation purposes. Six building models (Models *A* – *F*) were developed for training and testing of QTO algorithms. Three models (Models *G* – *I*) were utilized in evaluating the accuracy and robustness of the D-READ method.

Training and testing data: Three models were created based on the same apartment complex building project in Kalamazoo, Michigan. Hard copy architectural drawings were obtained from the project owner and 3D models of the building were created by the authors in three different BIM authoring tools according to the drawings, namely, Revit (Model *A*) (Fig. 6a), SketchUp (Model *C*) (Fig. 6b), and ArchiCAD (Model *E*) (Fig. 6c). The 3D model data were further converted into IFC format through the built-in exportation functions in the BIM authoring tools. Models *A, C, and E* were used for training. The testing Models *B, D, and F*, are shown in Figs. 6d, 6e, 6f, respectively. Model *B* was based on a residential building model retrieved from an online source Maro Design (2018) created in Autodesk Revit. Model *D* was based on a residential building created by Razin Kahn in Trimble SketchUp, retrieved from the online 3D Warehouse.

Model *F* was based on a residential building created in GRAPHISOFT ArchiCAD.

Evaluation data: Fig. 7 shows the three BIM instance models developed for evaluation purpose. Similar to the training and testing data, the three models were created in the three BIM authoring tools. Models G (Fig. 7a) and I (Fig. 7c) were created by the authors using Autodesk Revit and GRAPHISOFT ArchiCAD while Model *H* (Fig. 7b) was retrieved from the online 3D Warehouse based on a residential apartment building created by Razin Kahn in Trimble SketchUp.

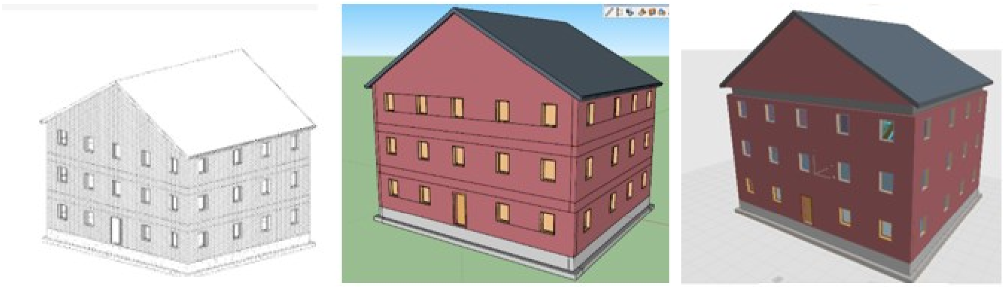


Fig. 6a. Model *A* **Fig. 6b**. Model *C* **Fig. 6c**. Model *E*

Fig. 6d. Model *B* **Fig. 6e**. Model *D* **Fig. 6f**. Model *F*

Fig. 6. Visualization of the Training and Testing Data.

Fig. 7a. Model *G* **Fig. 7b.** Model *H* **Fig. 7c.** Model *I*

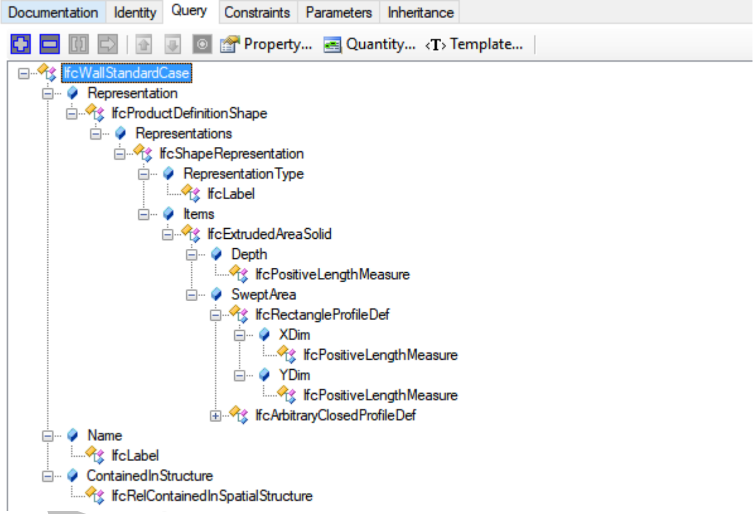
Fig. 7. Visualization of the Evaluation Data

Step 2: Model View Definition Development

To develop the MVD, the authors utilized the ifcDOC tool (buildingSMART 2015), which is an open-source MVD creation tool for creating model views, concept templates, concept roots, and concept leafs (Fig. 8).

In the scope identification sub-step, the main application was QTO. The other related application would usually be a BIM authoring tool (i.e., mostly an architectural BIM tool), or other BIM sources that may provide models for QTO purposes. This MVD was developed based on the CV V2.0, which has been developed by buildingSMART and supports IFC 2X3 exchange requirements in the areas of architectural exchange, structural exchange and building services exchange. In the exchange requirements definition sub-step, using the ifcDOC tool and the existing CV V2.0, the authors defined rules for supporting the QTO exchange requirements needed for exporting the IFC 2X3 file corresponding to a subset of the CV V2.0 from an architectural BIM tool. Four MVD concepts were defined including wall, floor, stair and roof. In the IFC entity mapping sub-step, the MVD concepts were mapped into IFC entities, together with attributes and constraints. As an example, in Fig. 8, an exchange requirement was defined for an

MVD concept wall, which in turn maps into the IFC entity *IfcWallStandardCase* with mandatory attributes "*Representation*," "*Name,*" and "*ContainedInStructure*;" the "*Representation*" attribute uses *IfcProductDefinitionShape*, which in turn, uses *IfcShapeRepresentation* and the *IfcShapeRepresentation* further uses *IfcExtrudedAreaSolid*. The *IfcExtrudedAreaSolid* is used to represent the details of a 3D shape, from which the needed geometric parameters for QTO can be found. The developed MVD validates if entities required for QTO correctly exist in the IFC instance files to ensure a successful QTO algorithm execution.



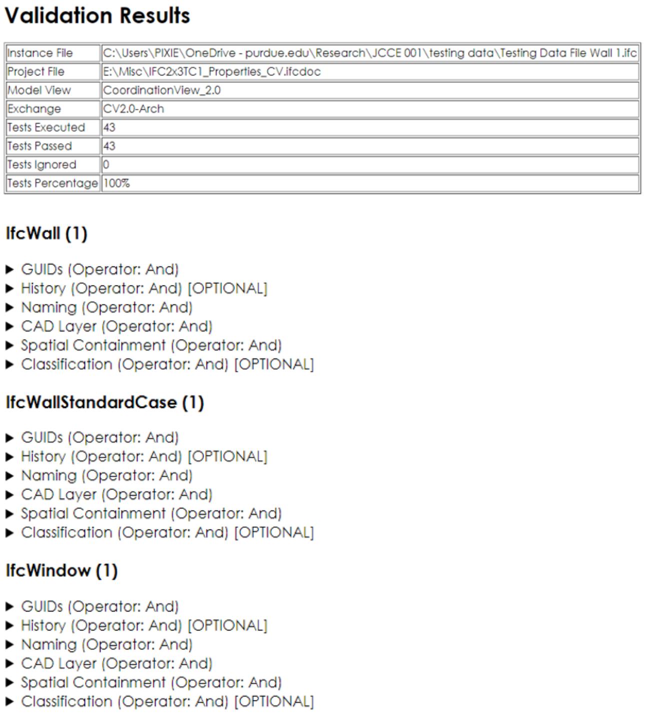
- IfcWallStandardCase
  - Representation
    - IfcProductDefinitionShape
      - Representations
        - IfcShapeRepresentation
          - RepresentationType
            - IfcLabel
          - Items
            - IfcExtrudedAreaSolid
              - Depth
                - IfcPositiveLengthMeasure
              - SweptArea
                - IfcRectangleProfileDef
                  - XDim
                    - IfcPositiveLengthMeasure
                  - YDim
                    - IfcPositiveLengthMeasure
                - IfcArbitraryClosedProfileDef
  - Name
    - IfcLabel
  - ContainedInStructure
    - IfcRelContainedInSpatialStructure

Fig. 8. Exchange Requirement for an MVD Concept Wall in the ifcDOC Interface

Fig. 9 shows an example HTML validation report generated from the MVD-based checking of a wall from the Model *B* IFC file. The results showed there were (1) an instance of *IfcWall*, (2) an instance of *IfcWallStandardCase*, and (3) an instance of *IfcWindow* in the IFC file. Two scenarios could arise from the validation results using this developed MVD: (1) insufficient information (as shown in Fig. 2, the system would inform the users to provide the needed information); or (2) sufficient information (the IFC file is passed on to the developed

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). “Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM.” J. Comput. in Civ. Eng., 34(5), 04020036.

QTO algorithms for data extraction). The validation results were extracted using jsoup Java HTML parser API (jsoup 1.11.3).



Validation Results

Instance File	C:\Users\PIXIE\OneDrive - purdue.edu\Research\JCCE 001\testing data\Testing Data File Wall 1.ifc
Project File	E:\Misc\IFC2x3TC1_Properties_CV.ifcdoc
Model View	CoordinationView_2.0
Exchange	CV2.0-Arch
Tests Executed	43
Tests Passed	43
Tests Ignored	0
Tests Percentage	100%

IfcWall (1)

- ▶ GUIDs (Operator: And)
- ▶ History (Operator: And) [OPTIONAL]
- ▶ Naming (Operator: And)
- ▶ CAD Layer (Operator: And)
- ▶ Spatial Containment (Operator: And)
- ▶ Classification (Operator: And) [OPTIONAL]

IfcWallStandardCase (1)

- ▶ GUIDs (Operator: And)
- ▶ History (Operator: And) [OPTIONAL]
- ▶ Naming (Operator: And)
- ▶ CAD Layer (Operator: And)
- ▶ Spatial Containment (Operator: And)
- ▶ Classification (Operator: And) [OPTIONAL]

IfcWindow (1)

- ▶ GUIDs (Operator: And)
- ▶ History (Operator: And) [OPTIONAL]
- ▶ Naming (Operator: And)
- ▶ CAD Layer (Operator: And)
- ▶ Spatial Containment (Operator: And)
- ▶ Classification (Operator: And) [OPTIONAL]

Fig. 9. An Output Report from an Example MVD Validation

Step 3: Data Analysis

Object Identification and Geometric Representation Analysis

Each object in the IFC files of the training models (Models *A, C, and E*) was identified and analyzed for their fundamental geometric representations in the IFC-based BIM. In total 61 objects were analyzed to identify tracing patterns for seven types of objects in the use of IFC entities and attributes.

In addition to the tracing pattern in the use of IFC data for the geometric representations of a rectangular wall (Fig. 4), tracing pattern of a curved wall was also analyzed to extract: (1) the radius of its center curve (*Radius1*); (2) the radius of its inner curve (*Radius2*); (3) the radius of its outer curve (*Radius3*); (4) the thickness of the wall(*WallDepth*); (5) the end point of the wall (*TrimOne*); and (6) the starting point of the wall (*TrimTwo*). These parameters are used in Equation [1] and Equation [2] to calculate the length (*L*) and width (*W*) of the wall, respectively. Fig. 10 below shows the detail of this tracing pattern. Similarly, the tracing patterns of a floor and a roof were analyzed to extract the corresponding parameters. For stairs, two different tracing patterns were found and analyzed. The first tracing pattern was used by Models A and C (Pattern S_1) (Fig. 11) and the second tracing pattern was used by Model E (Pattern S_2) (Fig. 12). The tracing pattern S_1 of the stairs was analyzed to extract: (1) the height of the riser (*RiserHeight*); (2) the depth of the thread (*ThreadDepth*); (3) the number of risers (*RiserNumber*); and (4) the number of threads (*ThreadNumber*). These parameters are used in Equation [3] to calculate the length of the flight (*FlightLength*). The tracing pattern S_2 of the stairs was analyzed to extract: (1) the height of the riser (*RiserHeight*); (2) the number of risers (*RiserNumber*); and (3) the number of threads (*ThreadNumber*). These parameters are used in Equations [3] and [4] to calculate the length of the flight *(FlightLength).* The main difference between these two tracing patterns of stairs are in the parameters used. While the *RelDefinesByProperties* in Pattern S_1 contained *ThreadDepth,* the *RelDefinesByProperties* in Pattern S_2 does not contain *ThreadDepth*.

$$L = 2\ X\ Pi\ X\ Radius1\ X\ \{1 - [\frac{TrimOne - TrimTwo}{360}]\} \qquad [1]$$

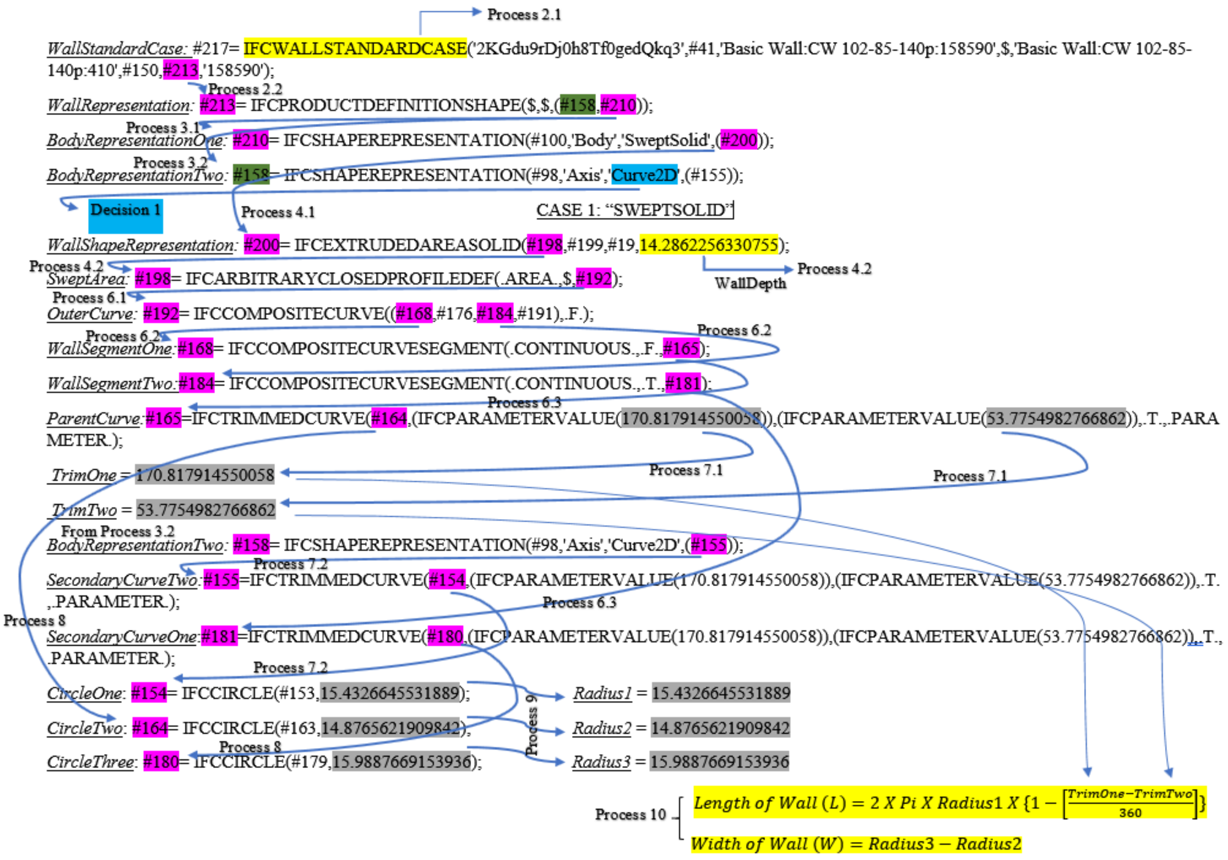
$$W = Radius3 - Radius2 \qquad [2]$$

$$FlightLength = ThreadNumber\ \mathrm{X}\ ThreadDepth \qquad [3]$$

$$ThreadDepth = 17.5\text{''} - RiserHeight \qquad [4]$$

The window and door tracing patterns were not considered separately but included as part of the wall tracing patterns and taken into account as openings.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.



```
WallStandardCase: #217= IFCWALLSTANDARDCASE('2KGdu9rDj0h8Tf0gedQkq3',#41,'Basic Wall:CW 102-85-140p:158590',$,'Basic Wall:CW 102-85-140p:410',#150,#213,'158590');
WallRepresentation: #213= IFCPRODUCTDEFINITIONSHAPE($,$,(#158,#210));
BodyRepresentationOne: #210= IFCSHAPEREPRESENTATION(#100,'Body','SweptSolid',(#200));
BodyRepresentationTwo: #158= IFCSHAPEREPRESENTATION(#98,'Axis','Curve2D',(#155));
CASE 1: "SWEPTSOLID"
WallShapeRepresentation: #200= IFCEXTRUDEDAREASOLID(#198,#199,#19,14.2862256330755);
SweptArea: #198= IFCARBITRARYCLOSEDPROFILEDEF(.AREA.,$,#192);
OuterCurve: #192= IFCCOMPOSITECURVE((#168,#176,#184,#191),.F.);
WallSegmentOne:#168= IFCCOMPOSITECURVESEGMENT(.CONTINUOUS.,.F.,#165);
WallSegmentTwo:#184= IFCCOMPOSITECURVESEGMENT(.CONTINUOUS.,.T.,#181);
ParentCurve:#165=IFCTRIMMEDCURVE(#164,(IFCPARAMETERVALUE(170.817914550058)),(IFCPARAMETERVALUE(53.7754982766862)),.T.,.PARAMETER.);
TrimOne = 170.817914550058
TrimTwo = 53.7754982766862
BodyRepresentationTwo: #158= IFCSHAPEREPRESENTATION(#98,'Axis','Curve2D',(#155));
SecondaryCurveTwo:#155=IFCTRIMMEDCURVE(#154,(IFCPARAMETERVALUE(170.817914550058)),(IFCPARAMETERVALUE(53.7754982766862)),.T.,.PARAMETER.);
SecondaryCurveOne:#181=IFCTRIMMEDCURVE(#180,(IFCPARAMETERVALUE(170.817914550058)),(IFCPARAMETERVALUE(53.7754982766862)),.T.,.PARAMETER.);
CircleOne: #154= IFCCIRCLE(#153,15.4326645531889);
CircleTwo: #164= IFCCIRCLE(#163,14.8765621909842);
CircleThree: #180= IFCCIRCLE(#179,15.9887669153936);
Radius1 = 15.4326645531889
Radius2 = 14.8765621909842
Radius3 = 15.9887669153936
```

$$\text{Length of Wall } (L) = 2\ X\ Pi\ X\ Radius1\ X\ \{1 - \left[\frac{TrimOne - TrimTwo}{360}\right]\}$$

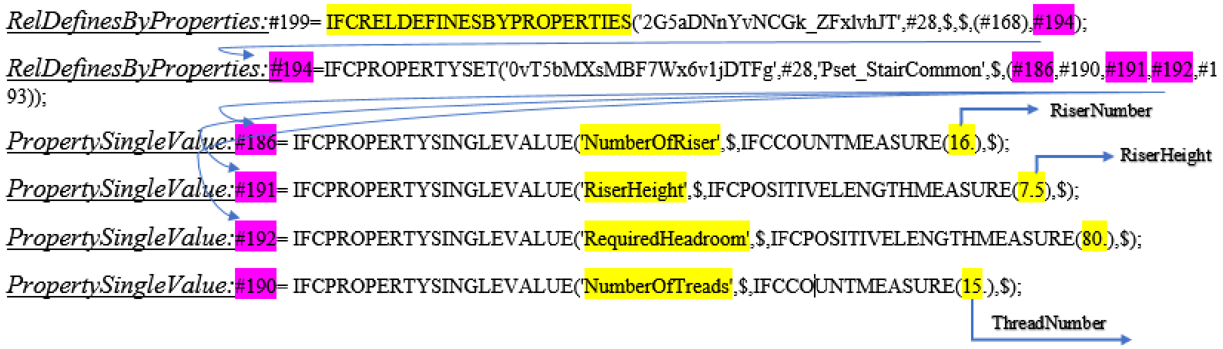
$$\text{Width of Wall } (W) = Radius3 - Radius2$$

Fig. 10. Tracing Pattern of a "SweptSolid" Representation of a Curved Wall

```
RelDefinesByProperties:#552= IFCRELDEFINESBYPROPERTIES('1QdVu3M0X1YQ6gic1h2bIY',#41,$,$,(#142),#528);
RelDefinesByProperties:#556= IFCRELDEFINESBYPROPERTIES('1IdEdE1a5FjwDfCxzk3KAp',#41,$,$,(#438),#542);
RelDefinesByProperties:#560= IFCRELDEFINESBYPROPERTIES('2IzjyK16XFZer7xRzZDSso',#41,$,$,(#486),#547);
RelDefinesByProperties:#564= IFCRELDEFINESBYPROPERTIES('0kxo9eD017VA6GYyURecB3',#41,$,$,(#519),#550);

PropertySingleValue:#528=IFCPROPERTYSET('1PKGTmvoLFjRy0$YJycLri',#41,'Pset_StairCommon',$,(#522,#523,#524,#525,#526,#527));
PropertySingleValue:#524= IFCPROPERTYSINGLEVALUE('NumberOfRiser',$,IFCCOUNTMEASURE(8.),$);
PropertySingleValue:#525= IFCPROPERTYSINGLEVALUE('NumberOfTreads',$,IFCCOUNTMEASURE(7.),$);
PropertySingleValue:#526=IFCPROPERTYSINGLEVALUE('RiserHeight',$,IFCPOSITIVELENGTHMEASURE(0.499999999999922),$);
PropertySingleValue:#527= IFCPROPERTYSINGLEVALUE('TreadLength',$,IFCPOSITIVELENGTHMEASURE(1.),$);
```

Fig. 11. Tracing Pattern S_1 of Stairs



```
RelDefinesByProperties:#199= IFCRELDEFINESBYPROPERTIES('2G5aDNnYvNCGk_ZFxlvhJT',#28,$,$,(#168),#194);
RelDefinesByProperties:#194=IFCPROPERTYSET('0vT5bMXsMBF7Wx6v1jDTFg',#28,'Pset_StairCommon',$,(#186,#190,#191,#192,#193));
PropertySingleValue:#186= IFCPROPERTYSINGLEVALUE('NumberOfRiser',$,IFCCOUNTMEASURE(16.),$);
PropertySingleValue:#191= IFCPROPERTYSINGLEVALUE('RiserHeight',$,IFCPOSITIVELENGTHMEASURE(7.5),$);
PropertySingleValue:#192= IFCPROPERTYSINGLEVALUE('RequiredHeadroom',$,IFCPOSITIVELENGTHMEASURE(80.),$);
PropertySingleValue:#190= IFCPROPERTYSINGLEVALUE('NumberOfTreads',$,IFCCOUNTMEASURE(15.),$);
```

Fig. 12. Tracing Pattern S_2 of Stairs

Step 4: Algorithm Development

The algorithms developed made use of tracing patterns in the IFC data structure and the component’s geometric representations in the IFC training data to generate the needed QTO. For example, the algorithms were developed to extract the height, length & width of a wall (rectangular or curved), and the height, length & width attributes of all its openings from its geometric representations in an IFC file. An example QTO algorithm for taking off the quantities of a rectangular wall is illustrated below.

Rectangular Wall - extracting the length, width and height attributes

Fig. 13 shows the partial QTO algorithm developed for a rectangular wall. Fig. 13a shows the part of the algorithm for extracting the height, length & width attributes of the rectangular wall; while Fig. 13b shows the part of the algorithm for extracting the height, length & width attributes of all of its openings from its geometric representation in an IFC file.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

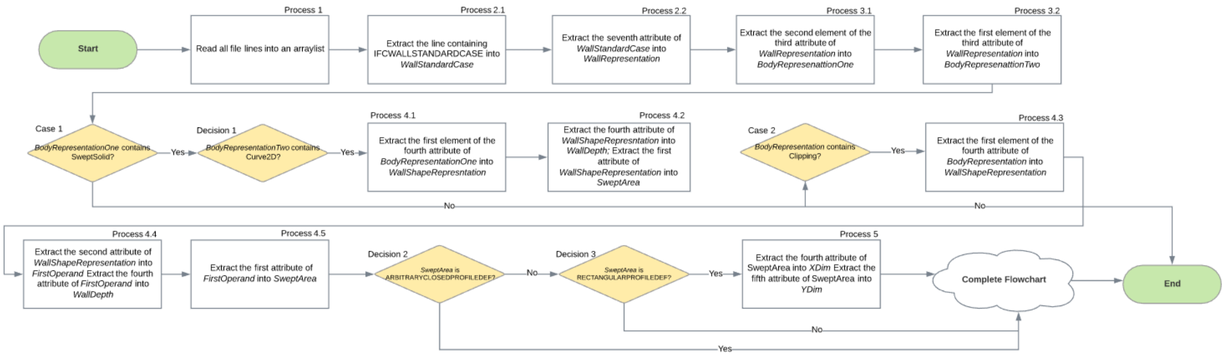


Fig. 13a. Partial Flow Chart of the QTO Algorithm for a Rectangular Wall

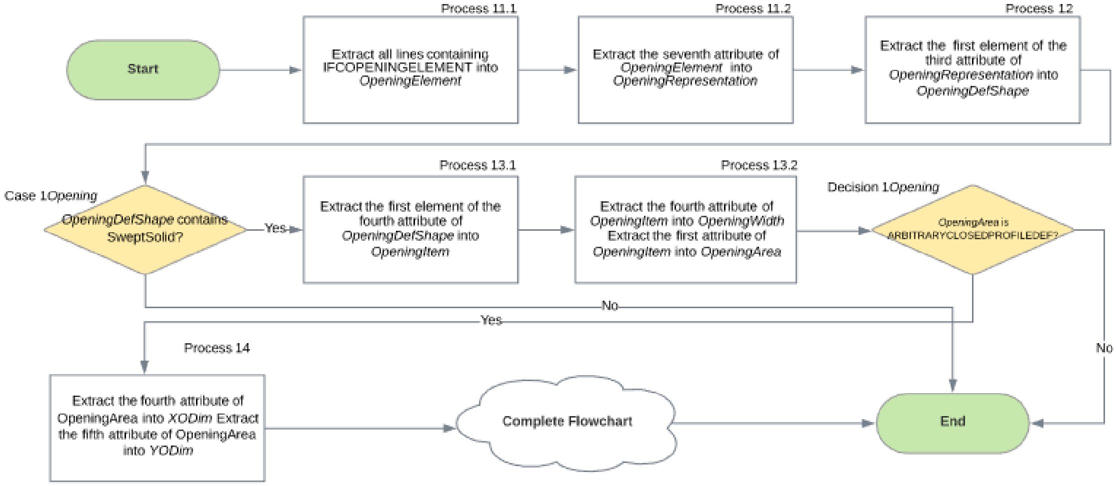


Fig. 13b. Partial Flow Chart of the QTO Algorithm for the Openings of a Rectangular Wall

Step 5: Algorithm Implementation and Testing

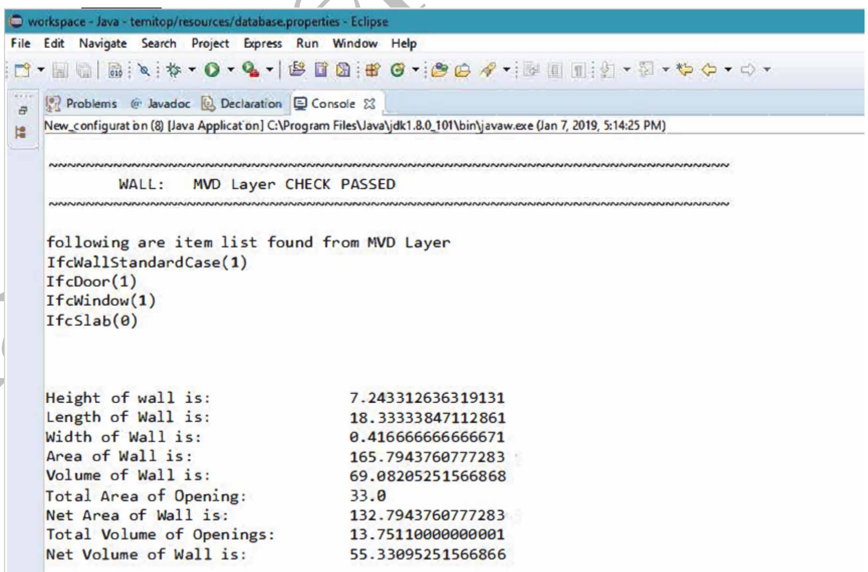
The developed QTO algorithms were tested in generating the QTO for objects in the three testing models (Models *B, D and F*), which were created in three different BIM authoring tools. In this way, we can evaluate the performance of the D-READ method in generating QTO for models using different authoring tools/workflows. One example object of each type was selected from each of the building models and the QTO results were tabulated in Table 1. Fig. 14 shows an

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

interface of output results of a wall instance (Model *B*) using the implemented QTO algorithms in Java.

Table 1. Testing results of selected objects

Component	Model	Length (ft.)	Width (ft.)	Height (ft.)	Area (sq.ft.)	Volume (cu.ft.)
Wall 1	B	18.3333	0.4167	7.2433	132.7944	55.3310
Wall 2	D	3.7989	0.625	29.2561	111.1408	69.463
Wall 3	F	3.655	0.4922	8.0	29.24	14.3919
Floor 1	B	26.3333	9.125	1.250	240.2914	300.3642
Floor 2	D	23.2550	18.5	0.9125	430.2175	392.5735
Floor 3	F	22.4269	8.780	0.8725	196.9082	171.8024
		Length (ft.)	**Width (ft.)**	**Area (sq. ft.)**	**Volume (cu. ft.)**	**Slope (^{0})**
Roof 1	B	25.3550	10.82	274.3411	1667.4452	-
Roof 2	D	53.0990	47	4991.306	30337.1579	30
Roof 3	F	21.3143	8.280	176.4824	926.5326	-
		Riser height (ft.)	**Thread Length (ft.)**	**Stairs width (ft.)**	**Flight Length (ft.)**	
Stairs 1	B	-	-	-	-	
Stairs 2	D	0.5577	0.7546	3.0348	18.62	
Stairs 3	F	-	-	-	-	



```
New_configurat on (8) [Java Applicat on] C:\Program Files\Java\jdk1.8.0_101\bin\javaw.exe (Jan 7, 2019, 5:14:25 PM)

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
        WALL:   MVD Layer CHECK PASSED
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

following are item list found from MVD Layer
IfcWallStandardCase(1)
IfcDoor(1)
IfcWindow(1)
IfcSlab(0)



Height of wall is:              7.243312636319131
Length of Wall is:              18.33333847112861
Width of Wall is:               0.416666666666671
Area of Wall is:                165.7943760777283
Volume of Wall is:              69.08205251566868
Total Area of Opening:          33.0
Net Area of Wall is:            132.7943760777283
Total Volume of Openings:       13.75110000000001
Net Volume of Wall is:          55.33095251566866
```

Fig. 14. An Example Output Result from a Wall Instance Using the Implemented QTO Algorithms in Java

Experimental Evaluation Results and Discussion

Accuracy of results: To evaluate the accuracy of the QTO algorithms produced by D-READ method, a comparison of the quantities obtained for Models *G and H* with those from a state-of-the-art estimating tool (Autodesk Navisworks) were recorded. As at the time of this research, the Autodesk Navisworks the authors had access to could only support models G and H. In contrast, the QTO algorithms developed using the D-READ method was successfully used in extracting the quantities from all the three models. A measurement of deviations between the results achieved using the commercial software (if the commercial software were able to provide the results), and the results achieved using the proposed method were tabulated in Table 2 and Table 3. The comparison of the tabulated quantities in these two tables shows that the proposed method and developed algorithms provided consistent results with that from the state-of-the-art commercial software, if the commercial software were able to provide the results.

Table 2. Accuracy of results (Model G)

	Method	Length (ft.)	Width (ft.)	Height (ft.)	Area (sq. ft.)	Volume (cu. ft.)
Wall 1						
	Algorithm	13.0577	0.4167	7.4602	97.4134	40.5922
	Commercial Software	13.0577	0.4167	7.4602	97.4134	40.5922
	Deviation (%)			0%		
Wall 2						
	Algorithm	20.6693	0.4167	7.9193	163.6868	68.2083
	Commercial Software	20.6693	0.4167	7.9193	163.6868	68.2083
	Deviation (%)			0 %		
Wall 3						
	Algorithm	20.21	0.4167	7.1768	145.0437	60.4397
	Commercial Software	20.21	0.4167	7.1768	145.0437	60.4397
	Deviation (%)			0 %		
Wall 4						
	Algorithm	6.7453	0.4167	7.4602	50.3213	20.9689
	Commercial Software	6.7453	0.4167	7.4602	50.3213	20.9689
	Deviation (%)			0 %		
Floor						

	Algorithm	39.0420	17.2498	0.6561	673.4656	441.8924
	Commercial Software	39.0420	17.2498	0.6561	673.4656	441.8924
	Deviation (%)			0 %		
		Length (ft.)	**Width (ft.)**	**Area (sq. ft.)**	**Volume (cu. ft.)**	**Slope (^{0})**
Roof						
	Algorithm	47.0	0.7916	2929.9364	2319.3349	30
	Commercial Software		0.7916	2929.9364	2319.3349	
	Deviation (%)			0 %		
		Riser height (ft.)	**Thread Length (ft.)**	**Flight Length (ft.)**	**Stairs width (ft.)**	
Stair						
	Algorithm	0.4261	0.8202	14.7638	2.9528	
	Commercial Software				2.9528	
	Deviation (%)			0 %		

Table 3. Accuracy of results (Model H)

	Method	**Length (ft.)**	**Width (ft.)**	**Height (ft.)**	**Area (sq. ft.)**	**Volume (cu. ft.)**
Wall 1						
	Algorithm	21.1942	0.4167	9.0914	159.6839	66.5350
	Commercial Software	21.1942	0.4167	9.0914	159.6839	66.5350
	Deviation (%)			0%		
Wall 2						
	Algorithm	20.3280	0.4167	8.8583	131.8486	54.9369
	Commercial Software	20.3280	0.4167	8.8583	131.8486	54.9369
	Deviation (%)			0 %		
Wall 3						
	Algorithm	20.7283	0.4167	8.8583	177.6169	74.0070
	Commercial Software	20.7283	0.4167	8.8583	177.6169	74.0070
	Deviation (%)			0 %		
Wall 4						
	Algorithm	14.5647	0.4167	8.8583	123.0178	51.2574
	Commercial Software	14.5647	0.4167	8.8583	123.0178	51.2574
	Deviation (%)			0 %		
Floor						
	Algorithm	29.3044	21	1.0625	615.3937	653.8558
	Commercial Software	29.3044	21	1.0625	615.3937	653.8558
	Deviation (%)			0 %		

Robustness of method:

To evaluate the robustness of the proposed D-READ method, a comparison of the D-READ method in generating QTO of components from various BIM authoring tools against different state-of-the-art commercial software were tabulated in Table 4. Three BIM instance models (Models *G, H and I*) were utilized in performing the robustness test. Three state-of-the-art estimating tools from the three parent-company of the used BIM authoring tools (i.e., Autodesk, Trimble and GraphiSOFT) were chosen to check if each tool can perform QTO on each of the three models. The three estimating tools were Autodesk Naviswork, Trimble GCEstimator, and GraphiSOFT ArchiCAD. As at the time of this research, the authors had access to only Autodesk Naviswork and GraphiSOFT ArchiCAD. The review of Trimble GCEstimator in supporting other formats was conducted via the software's support page. The results suggest that while the state-of-the-art software are not comprehensive in supporting the different BIM authoring tools, the D-READ method successfully developed QTO algorithms that extracted the quantities from models created in all the three BIM authoring tools.

Table 4. Robustness evaluation

	QTO tool compatibility			
BIM platform	D-READ	Autodesk Naviswork	Trimble GCEstimator	GraphiSOFT ArchiCAD
Autodesk Revit (Model G)	Yes	Yes	No	Yes
Trimble SketchUp (Model H)	Yes	Yes	Yes	No
GraphiSOFT ArchiCAD (Model I)	Yes	No	No	Yes
Other BIM Platforms	Yes	?	?	?

Contributions to the body of knowledge

This research contributes to the body of knowledge in two main ways: First, this study brings a new data-driven method for automated QTO algorithm development using IFC-based BIMs. The proposed method leveraged model-specific geometric representations of building components in an IFC file directly. In contrast to existing BIM-based QTO methods that only deal with specific/selected/proprietary BIM-authoring tool/workflow, this method can be utilized to develop QTO algorithms that can be applied to models created from any IFC-compatible BIM authoring tools/workflows. This is more robust than workflows built on proprietary data formats and therefore can provide QTO algorithms with a higher level of support to BIM interoperability. The QTO algorithms developed can be accumulatively grown into a comprehensive set to cover different objects in different types of construction projects. Second, the presented work extended the current available architectural MVD specifications to one that checks IFC instance files for architectural QTO purposes. This is a pioneer research effort in systematically solving interoperability and automation of BIM-based QTO.

Conclusions, Limitations and Future Work

To establish interoperable QTO methods using BIMs created from different BIM authoring tools/workflows, the authors developed a new D-READ method that can be applied to develop algorithms for extracting the needed quantities from any building object by leveraging the geometric shape representations of the objects in an IFC model. The proposed method was tested using nine BIM instance models from three different BIM authoring tools/workflows – three for training, three for testing, and three for evaluation. QTO algorithms were produced for

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

wall, roof, floor, and stairs as a result of applying the proposed method. These produced QTO algorithms were applied to the evaluation models to test their accuracy and robustness, in comparison with state-of-the-art QTO tools. The algorithms successfully extracted the quantities of the evaluation models in consistent with the state-of-the-art tools. While none of the state-of-the-art tools could successfully process all the different evaluation models because of their different sources and therefore the different uses of IFC entities/attributes, the developed QTO algorithms were able to achieve that. The D-READ method proposed in this study therefore establishes an approach that can be applied to the development of QTO algorithms of building components using a broad range of IFC-based BIMs (e.g., by different BIM authoring tools/workflows) to support BIM interoperability.

One main limitation is acknowledged: currently the D-READ method produces QTO algorithms that could only address geometric representations that were observed in the training data. A boosting strategy will be investigated in future research to enable the D-READ method to cover a broader set of IFC data patterns than those observed in training data.

Acknowledgement

The authors would like to thank the National Science Foundation (NSF). This material is based upon work supported by NSF under Grant No. 1745374. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

References

buildingSMART. (2011). "An Integrated Process for Delivering IFC Based Data Exchange." < https://www.standard.no/Global/PDF/ISO-TC59-SC13/N_287_Integrated_IDM-MVD_Process_for_IFC-formats.pdf> (June. 25, 2019).

buildingSMART. (2015). "ifcDoc Tool Summary." <http://www.buildingsmart-tech.org/specifications/specification-tools/ifcdoc-tool/ifcdoc-beta-summary> (July 6th, 2015)

buildingSMART. (2019). "Software Compliance: Certified Software." < https://www.buildingsmart.org/compliance/software-certification/certified-software/> (June. 25, 2019).

Cheung, F., Rihan, J., Tah, J., Duce, D., and Kurul, E. (2012). "Early stage multi-level cost estimation for schematic BIM models." *Automation in Construction*, 27(2012), 67-77.

Choi, J., Kim, H., and Kim, I. (2011). "Interoperability tests between IFC certified Software for Open BIM based quality assurance." *Proc., 28th International Symposium on Automation and Robotics in Construction, Seoul*, Korea, 231-244.

Choi, J., Kim, H., and Kim, I. (2015). "Open BIM-based quantity takeoff system for schematic estimation of building frame in early design stage." *Journal of Computational Design and Engineering,* 2(2015), 16-25.

Costin, A., and Eastman, C. (2019). "Need for interoperability to Enable Seamless Information Exchanges in Smart and Sustainable Urban Systems." *J. Comput. Civ. Eng.,* 2019, 33(3): 04019008

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Ding, L., Li, K., Zhou, Y., and Love, P. (2017). "An IFC-inspection process model for infrastructure projects: Enabling real-time quality monitoring and control." *Automation in Construction*, 84, 96-110.

Drogemuller, R. (2005). "Model based estimating system for civil concrete structures." Final Report 2005-0008-C-01, CRC for Construction Innovation, Queensland University of Technology, Brisbane City, Australia.

Drogemuller, R.M. (2003). "Managing information flows with models and virtual environments." Project Report, CRC-CI Report No. 2001-007-C-01, CRC for Construction Innovation, Queensland University of Technology, Brisbane City, Australia.

Eastman, C., Sacks, R., Panushev, I., Aram, S., and Yagmur, K. (2009). "Information Delivery Manual for Precast Concrete" < http://dcom.arch.gatech.edu/pcibim/documents/IDM_for_Precast.pdf> (Jan. 26, 2019).

Franz, B., and Messner, J. (2017). "Evaluating the impact of BIM on project performance." < https://www.bim.psu.edu/download/franz-and-messner-evaluating-the-impact-of-bim-on-project-performance-2018.pdf > (Jun. 15, 2019).

Golabchi, A., and Kamat, V (2013). "Evaluation of industry foundation classes for practical building information modeling interoperability." Proc., 30th International Symposium on Automation and Robotics in Construction, Montreal, Canada, 17-26.

Franco, J., Mahdi, F., and Abaza, H. (2015). "Using Building Information Modeling (BIM) for estimating and scheduling, adopting barriers." *Universal Journal of Management,* 9(2015), 376-384.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Hernandez, J., Lerones, P., Bonsma, P., Delft, A., Deighton, R., and Braun, J. (2018). "An IFC Interoperability Framework for Self-Inspection Process in Buildings." *Buildings* 2018, 8(2), 32.

Jsoup Java HTML Parser 1.11.3. [Computer Software]. (<https://jsoup.org/news/release-1.11.3>), 2018.

Khan, R. (2015). "American system built homes 2." <https://3dwarehouse.sketchup.com/model/uaac930c6-9933-4c54-9df8-831a84756b1f/American-system-built-homes-2> (Jun. 26, 2019).

Khan, R. (2016). "Darwin D. Martin Gardener's Cottage." <https://3dwarehouse.sketchup.com/model/44562ca4-d01c-4038-8c49-bb067cf92d40/Darwin-D-Martin-Gardeners-Cottage> (Jun. 26, 2019).

Kreider, R., Messner, J., and Dubler, C. (2010). "Determining the frequency and impact of applying BIM for different purposes on building projects." *Proc., 6th Int. Conf. on Innovation in Architecture, Engineering and Construction (AEC)*, Pennsylvania State Univ., University Park, PA, USA.

Lai, H., and Deng, X. (2018). "Interoperability analysis of IFC-based data exchange between heterogeneous BIM software." *Journal of Civil Engineering and Management,* 24(7), 537-555.

Lee, J., Smith, J., and Kang, J. (2011). "The role of IFC for sustainable BIM data management." *Proc., 28th International Symposium on Automation and Robotics in Construction*, IAARC, Seoul, Korea, 764-769.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Lee, S., Kim, K., and Yu, J. (2014). "BIM and ontology-based approach for building cost estimation." *Automation in Construction,* 41(2014), 96-105.

Lee, Y., Eastman C., and Lee, J. (2015). "Validations for ensuring the interoperability of data exchange of a building information model." *Automation in Construction,* 58(2015), 176-195.

Lee, Y., Eastman C., and Solihin, W. (2016). "An ontology-based approach for developing data exchange requirements and model views of building information modeling." *Advanced Engineering Informatics,* 30(2016), 354-367.

Lee, Y., Eastman, C., Solihin, W. (2018). "Logic for ensuring the data exchange integrity of building information models." *Automation in Construction*, 85, 249-262.

Lee, Y., Solihin, W., Eastman, C. (2019). "The Mechanism and Challenges of Validating a Building Information Model regarding data exchange standards." *Automation in Construction*, 100, 118-128.

Liu, R., and Issa, R.R.A. (2013). "Issues in BIM for Facility Management from Industry Practitioners' Perspectives." Proc., International Workshop on Computing in Civil Engineering, 2013, ASCE, Reston, VA, 411-418.

Liu, R., and Issa, R.R.A. (2014). "Design for maintenance accessibility using BIM tools." Facilities, 32(3/4), 153-159.

Liu, R., and Issa, R.R.A. (2015). "Survey: Common Knowledge in BIM for Facility Maintenance." Journal of Performance of Constructed Facilities, 30(3), 04015033.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Ma, Z., Wei, Z., and Zhang, X. (2013). "Semi-automatic and specification-compliant cost estimation for tendering of building projects based on IFC data of design model." *Automation in Construction,* 30(2013), 126-135.

Maro Design. (2018). "Wood House 1 revit 3D model." <https://www.turbosquid.com/3d-models/wood-house-1-revit-3d-model-1313024> (Jan. 30, 2019).

McGraw-Hill Construction. (2014). "The business value of BIM in construction for major global markets: How contractors around the world are driving innovation with building information modeling." Bedford, MA.

Monteiro, A., and Martins, J. (2015). "A survey on modeling guidelines for quantity takeoff-oriented BIM-based design." *Automation in Construction,* 35(2013), 238-253.

Mutis, I., and Paramashivam, A. (2019). "Cybersecurity management framework for a cloud-based BIM model." *I. Mutis and T. Hartmann (eds.), Advances in Informatics and Computing in Civil and Construction Engineering,* Springer Nature Switzerland AG 2019, Basel, Switzerland, 325-333.

Mutis, I., Ramachandran. A., and Martinez, M.G. (2019). "The BIMbot: A cognitive assistant in the BIM room." I. Mutis and T. Hartmann (eds.), *Advances in Informatics and Computing in Civil and Construction Engineering*, Springer Nature Switzerland AG 2019, Basel, Switzerland, 155-163.

Nawari, N. (2012). "BIM standardization and wood structures." *Proc., ASCE International Conference on Computing in Civil Engineering,* ASCE, Reston, VA, 293-300.

Plebankiewicz, E., Zima, K., and Skibniewski, M. (2015). "Analysis of the first polish BIM-Based cost estimation application." *Procedia Engineering,* 123(2015), 405-414.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Ramaji, I., and Memari, A. (2018a). "Interpretation of structural analytical models from the coordination view in building information models." Automation in Construction, 90(2018), 117-133.

Ramaji, I., and Memari, A. (2018b). "Extending the current model view definition standards to support multi-storey modular building projects." *Architectural Engineering and Design Management,* 14(1-2), 158-176.

Santos, A., Costa, A., and Grillo, A. (2017). "Bibliometric analysis and review of Building Information Modeling literature published between 2005 and 2015." *Automation in Construction,* 80(2017), 118-136.

Samphaongoen P. (2010). "A visual approach to construction cost estimating." Master's Theses. Marquette University, Milwaukee, Wisconsin. Paper 28 <https://epublications.marquette.edu/theses_open/28> (May 28, 2017).

Sun, J., Liu, Y-S., Gao, G., and Han, X-G. (2015). "IFCCompressor: A content-based compression algorithm for optimizing Industry Foundation Classes files" Automation in Construction, 50 (2015), 1-15.

Steel, J., Drogemuller, R. & Toth, B. (2012). "Model Interoperability in Building Information modelling." Softw. Syst. Model, 11(1), 99-109.

Venugopal, M., Eastman, C., Birgonul, T., Sacks, R., and Teizer, J. (2012). "Semantics of model views for information exchanges using the industry foundation class schema." *Adv. Eng. Inform,* 26 (2) (2012), 419-432.

Wu, J., and Zhang, J. (2019). "A new automated BIM object classification method to support BIM interoperability." J. Comput. Civ. Eng., in press.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). "Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM." J. Comput. in Civ. Eng., 34(5), 04020036.

Weise, M., Liebich, T., and Wix, J. (2008). "Integrating use case definitions for IFC developments." in RS. Alain Zarli (Ed.), *ECPPM 2008: eWork and eBuisness in Architecture, Engineering and Construction*, Taylor & Francis, 2008, 637-645.

Yu, W., Lai, C., and Lee, W. (2006). "A WICE approach to real-time construction cost estimation." *Automation in Construction,* 15(2006), 12-19.

Zhang, J., and El-Gohary, N. (2015a). "Automated extraction of information from building information models into a semantic logic-based representation." Proc., 2015 ASCE Intl. Workshop on Comput. in Civ. Eng., ASCE, Reston, VA, 173-180.

Zhang, J., and El-Gohary, N. (2015b). "Automated information transformation for automated regulatory compliance checking in construction." J. Comput. in Civ. Eng., 29(4), B4015001.

Zhang, J., and El-Gohary, N. (2013). "Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking." J. Comput. in Civ. Eng., 30(2), 04015014.

Zhang, J. (2018). "Towards systematic understanding of geometric representations in BIM standard: an empirical data-driven approach." Proc., ASCE Construction Research Congress, ASCE, Reston, VA 96-105.

Zhang, S., Sulankivi, K, Kiviniemi, M., Romo, I., Eastman, C.M., and Teizer, J. (2015). "BIM-based fall hazard identification and prevention in construction safety planning." Safety Science, 72(February), 31-45.

Figure Captions List

Fig. 1. Proposed Data-Driven Reverse Engineering Algorithm Development (D-READ) Method

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). “Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM.” J. Comput. in Civ. Eng., 34(5), 04020036.

Fig. 2. An Illustration of the Application of Developed QTO Algorithms

Fig. 3. Object Identification Processes

Fig. 4. Tracing Pattern of a “SweptSolid” Representation of a Rectangular Wall

Fig. 5. UML Class Diagram for the Algorithm Development

Fig. 6. Visualization of the Training Data and Testing Data

Fig. 7. Visualization of the Evaluation Data

Fig. 8. Exchange Requirement for an MVD Concept Wall in the ifcDOC Interface

Fig. 9. An Output Report from an Example MVD Validation

Fig. 10. Tracing Pattern of a “SweptSolid” Representation of a Curved Wall

Fig. 11. Tracing Pattern S_1 of Stairs

Fig. 12. Tracing Pattern S_2 of Stairs

Fig. 13a. Partial Flow Chart of the QTO Algorithm for a Rectangular Wall

Fig.13b. Partial Flow Chart of the QTO Algorithm for the Openings of a Rectangular Wall

Fig. 14. An Example Output Result from a Wall Instance Using the Implemented QTO Algorithms in Java