

**IFC-BASED SYSTEMS AND METHODS
TO SUPPORT CONSTRUCTION COST ESTIMATION**

by

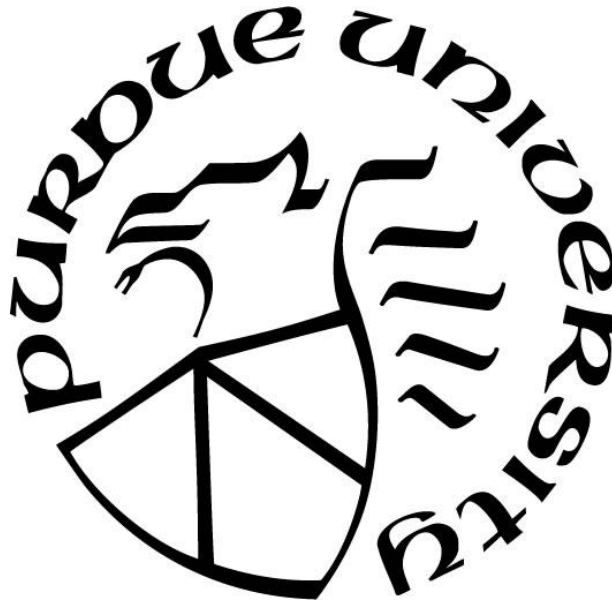
Temitope Akanbi

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy



Department of Technology

West Lafayette, Indiana

May 2021

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Jiansong Zhang, Chair

School of Construction Management Technology

Dr. Hubo Cai

Lyles School of Civil Engineering

Dr. Luciana Debs

School of Construction Management Technology

Dr. Anthony Sparkling

School of Construction Management Technology

Dr. Yi Jiang

School of Construction Management Technology

Approved by:

Dr. Kathryn Newton

*To my wife,
for the relentless love, support, encouragement, and patience
during the years it took me to finish my programs.*

ACKNOWLEDGMENTS

ADVISOR.

First and foremost, I would like to express my sincere acknowledgement to my advisor, Prof. Jiansong Zhang, for the opportunity to be part of his research group. It has been an honor to be his first Ph.D. student. This dissertation would not have been possible without his continuous support, immense knowledge, and guidance. Under his tutorship, I have learned the art of conducting research and technical writing.

COMMITTEE MEMBERS.

Besides my advisor, I would like to thank the rest of my dissertation committee members (Dr. Hubo Cai, Dr. Luciana De Cresce El Debs, Dr. Anthony Sparkling, and Dr. Jiang Yi) for their invaluable advice and support.

LAB MATES.

I would also like to acknowledge my AutoIC lab mates for their support and encouragement. This dissertation would not have been possible without the contributions from Oscar Wong Chong, Ran Ren, and Jin Wu.

INDUSTRY EXPERTS.

I would like to acknowledge the industry experts whose immense knowledge helped contribute to the success of this dissertation.

FAMILY AND FRIENDS.

Lastly, I would like to acknowledge my siblings, in-laws, cousins and friends for the financial support, endless love, emotional support and for always showing up.

TABLE OF CONTENTS

LIST OF TABLES	9
LIST OF FIGURES	10
LIST OF ABBREVIATIONS	13
PREFACE	14
ABSTRACT	15
CHAPTER 1. INTRODUCTION	17
1.1 Motivation and Overview	17
1.2 Background	17
1.3 Significance.....	18
1.4 Problem Statement	21
1.5 Research Questions	21
1.6 Assumptions.....	21
1.7 Limitations	21
1.8 Delimitations	22
1.9 Research Aims and Objectives	23
1.10 Outline of the Dissertation Structure	23
CHAPTER 2. LITERATURE REVIEW	25
2.1 Cost Estimation	25
2.1.1 Work Breakdown Structure (WBS) and Cost Breakdown Structure (CBS)	26
2.2 Building Information Modeling.....	28
2.2.1 Level of Development/Detail	28
2.2.2 Building Information Modeling in Vertical Construction	29
2.2.3 Building Information Modeling in Horizontal Construction	30
2.3 Industry Foundation Classes	31
2.3.1 Model View Definition.....	32
2.4 Natural Language Processing (NLP)	33
CHAPTER 3. MODELING COMPONENT - FRAMEWORK FOR DEVELOPING IFC-BASED 3D DOCUMENTATION FROM 2D DRAWINGS.....	38
3.1 3D Shape Generation	38

3.2	Proposed Component for Generating 3D Information Model	40
3.2.1	Step 1- PDF Import.....	41
3.2.2	Step 2- PDF File Conversion.....	41
3.2.3	Step 3- Raster Graphics Sheets Selection.....	42
3.2.4	Step 4- Raster Graphics File Conversion.....	42
3.2.5	Step 5- Vector Graphics File Conversion.....	44
3.2.6	Step 6- Tagged Data File Conversion.....	44
3.2.7	Step 7- IFC File Generation.....	49
3.3	Experimental Testing and Evaluation.....	49
3.3.1	Experimental Data	49
3.3.2	Evaluation	50
3.3.3	Experiment.....	50
3.3.4	Experimental Results and Discussion.....	60
3.4	Conclusions, Contributions, Limitations, and Recommendations for Future Research ...	64
3.4.1	Conclusions for the Proposed Modeling Component.....	64
3.4.2	Contributions of the Proposed Modeling Component.....	65
3.4.3	Limitations of the Proposed Modeling Component.....	65
CHAPTER 4. QUANTIFICATION COMPONENT – A DATA-DRIVEN REVERSE ENGINEERING ALGORITHM DEVELOPMENT (D-READ) METHOD FOR DEVELOPING INTEROPERABLE QUANTITY TAKEOFF ALGORITHM USING IFC-BASED BIM.....		67
4.1	IFC-Based QTO Methods	67
4.2	Proposed Method	68
4.2.1	Step 1- Model Development.....	70
4.2.2	Step 2- Model View Definition (MVD) Development.....	70
4.2.3	Step 3- Data Analysis	71
4.2.4	Step 4- Algorithm Development.....	80
4.2.5	Step 5- Algorithm Implementation & Testing.....	80
4.3	Experimental Testing and Evaluation.....	82
4.3.1	Experimental Data	82
4.3.2	Experimental Evaluation Results and Discussion	90
4.4	Conclusions, Contributions, Limitations, and Recommendations for Future Research ...	93

4.4.1	Conclusions for the Proposed Quantification Component	93
4.4.2	Contributions of the Proposed Quantification Component	94
4.4.3	Limitations of the Proposed Quantification Component	94
CHAPTER 5. EXTRACTION COMPONENT - AUTOMATED DESIGN INFORMATION EXTRACTION FROM SPECIFICATIONS TO SUPPORT CONSTRUCTION COST ESTIMATION AUTOMATION		95
5.1	Background	96
5.2	Proposed Method	97
5.2.1	Step 1 - Semantic Modeling	98
5.2.2	Step 2- Cost Database Creation	98
5.2.3	Step 3- Information Extraction and Matching Algorithm Development	100
5.2.4	Step 4: Database Iteration	104
5.2.5	Material Unit Price	105
5.3	Experimental Testing and Evaluation	105
5.3.1	Experimental Set-Up and Data	107
5.3.2	Experimental Results	112
5.4	Conclusions, Contributions, Limitations, and Recommendations for Future Research .	117
5.4.1	Conclusions for the Proposed Extraction Component	117
5.4.2	Contributions of the Proposed Extraction Component	118
5.4.3	Limitations of the Proposed Extraction Component	118
CHAPTER 6. COSTING COMPONENT - AUTOMATED ITEM MATCHING AND PRICING (IMP) FOR WOOD BUILDING ELEMENTS TO SUPPORT BIM-BASED WOOD CONSTRUCTION COST ESTIMATION		119
6.1	Background	119
6.2	Proposed Method for Automated Item Matching and Pricing (IMP)	120
6.3	Experimental Testing and Evaluation	121
6.3.1	Experimental Data	121
6.4	Conclusions, Contributions, Limitations, and Recommendations for Future Research .	127
6.4.1	Conclusions for the Proposed Costing Component	127
6.4.2	Contributions of the Proposed Costing Component	128
6.4.3	Limitations of the Proposed Costing Component	128

CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS	129
7.1 Summary and Conclusions	129
7.2 Comparison of the IFC-Based Framework with Current State-of-the-art Methods in Generating Cost Estimates	130
7.3 Research Contributions	133
7.4 Research Limitations	133
7.5 Recommendations for Future Research	134
REFERENCES	135
APPENDIX A – PERMISSION TO REUSE CONTENT	144
APPENDIX B – BRIDGE PLANS UTILIZED FOR DEVELOPING IFC-BASED 3D DOCUMENTATION FROM 2D PDF DRAWINGS	145
APPENDIX C – CONSTRUCTION DRAWINGS UTILIZED FOR DEVELOPING D-read METHOD FOR DEVELOPING INTEROPERABLE QUANTITY TAKEOFF ALGORITHM USING IFC-BASED BIM	146
APPENDIX D – 3D MODEL RENDERING UTILIZED FOR THE QUANTIFICATION COMPONENT	154
APPENDIX E – IFC FILES	155
APPENDIX F – CONSTRUCTION DRAWINGS UTILIZED FOR AUTOMATED DESIGN INFORMATION EXTRACTION FROM SPECIFICATIONS	163
APPENDIX G – SOURCE CODE- QUANTIFICATION COMPONENT	164
APPENDIX H – SOURCE CODE- EXTRACTION & COSTING COMPONENTS	173

LIST OF TABLES

Table 3.1. Experimental Results (Summary).....	63
Table 3.2. Partial Experimental Results (Calculated Deviation in Distances)	64
Table 4.1. Testing results of selected objects	89
Table 4.2. Accuracy of results (Model G)	91
Table 4.3. Accuracy of results (Model H)	92
Table 4.4. Robustness evaluation.....	93
Table 5.1. Gold Standard – Exterior Wall (Project C).....	110
Table 5.2. Number of Information Elements in Gold Standard Development – Project C	112
Table 5.3. Construction Sections Processed (Project C).....	114
Table 5.4. Experimental Results	114
Table 5.5. Experimental Results	116
Table 7.1. Step-by-Step Comparison of the State-of-the-Art Method and the Proposed Framework	11631

LIST OF FIGURES

Figure 1.1. Typical Steps in Cost Estimation using Software	18
Figure 1.2. IFC-Based Framework to Support Cost Estimation	20
Figure 2.1. Example MasterFormat Structure of Construction Specifications	27
Figure 2.2. Partial Image of Part 2 – Products (Division 072100 – Thermal Insulation) of an Example Specifications Document	36
Figure 2.3. Partial Image of Section 092900 – Gypsum Board (Part 1, Section 1.2, Paragraphs B) of an Example Specifications Document	37
Figure 3.1. Orthographic Views of a Bridge.....	39
Figure 3.2. Projections of a Bridge Structure	40
Figure 3.3. Proposed Framework for Generating 3D Information Model	41
Figure 3.4. Processes Involved in Converting the Raster Graphics File to a Vector Graphics File	43
Figure 3.5. Processes Involved in Converting the Tagged Data Graphics File Format to OBJ File Format	45
Figure 3.6. (a) Unaligned Projections of a Bridge; (b) Aligned Projections of a Bridge	46
Figure 3.7. Sub-step b: Creating Orthogonal Lines	47
Figure 3.8. Sub-step c: Match Coordinates (Unmatched Coordinates Highlighted in Red).....	48
Figure 3.9. BIM Renderings of the Bridges.....	50
Figure 3.10. Partial Flowchart of the 3D Model Generation Algorithms	52
Figure 3.11. Step 2 Algorithm, PDF File Conversion	53
Figure 3.12. Step 3 Algorithm, Raster Graphics File Selection.....	54
Figure 3.13. Step 4 Algorithm, Raster Graphics File Conversion	55
Figure 3.14. Step 5 Algorithm, Vector Graphics File Conversion	56
Figure 3.15. Projections of a Bridge Structure in a Scalable Vector Graphics File Format (Inkscape Graphics Editor).....	56
Figure 3.16. Projections of a Bridge Structure in DXF File Format.....	57
Figure 3.17. Step 6 Algorithm, DXF File Conversion.....	58
Figure 3.18. Examples of the Image of the OBJ Files Generated.....	58
Figure 3.19. Step 7 Algorithm, IFC File Generation	59

Figure 3.20. Partial Output (IFC File)	59
Figure 3.21. IFC Output Files Imported in BIMvision	60
Figure 3.22. CloudCompare Statistical Test Distribution Histogram.....	62
Figure 3.23. Cloud-to-Cloud distances (Example of Results – Bridge C).....	63
Figure 4.1. Proposed Data-Driven Reverse Engineering Algorithm Development (D-READ) Method	69
Figure 4.2. An Illustration of the Application of Developed QTO Algorithms	70
Figure 4.3. Object Identification Processes	72
Figure 4.4. Tracing Pattern of a “SweptSolid” Representation of a Rectangular Wall	73
Figure 4.5. Tracing Pattern of a “SweptSolid” Representation of a Curved Wall.....	75
Figure 4.6. Tracing Pattern of a “Clipping” Representation of a Rectangular Wall	76
Figure 4.7. Tracing Pattern of an Opening	77
Figure 4.8. Tracing Pattern of a “SweptSolid” Representation of a Rectangular Floor	78
Figure 4.9. Tracing Pattern of the Material Layer Attributes	79
Figure 4.10. Tracing Pattern S_1 of Stairs	80
Figure 4.11. Tracing Pattern S_2 of Stairs	80
Figure 4.12. UML Class Diagram for the Algorithm Development.....	81
Figure 4.13. Visualization of the Training and Testing Data	83
Figure 4.14. Visualization of the Evaluation Data.....	83
Figure 4.15. Exchange Requirement for an MVD Concept Wall in the ifcDOC Interface	85
Figure 4.16. An Output Report from an Example MVD Validation	86
Figure 4.17. Flowchart of the QTO algorithms for a Wall	87
Figure 4.18. Flowchart of the QTO algorithms for a Floor	88
Figure 4.19. An Example Output Results from a Wall Instance Using the Implemented QTO Algorithms in Java	89
Figure 5.1. Proposed Method.....	97
Figure 5.2 Partial Hierarchical Model of a Semantic Model	98
Figure 5.3. The Cost Database	99
Figure 5.4. Processes involved in the Cost Database Creation.....	99
Figure 5.5. Sub-Entity, Attributes & Values (Wood)	100

Figure 5.6. Processes Involved in Information Extraction and Matching Algorithm Development	100
Figure 5.7. Flow Chart of a Sample Information Extraction and Matching Algorithm	104
Figure 5.8. Revit Rendering of Project A	106
Figure 5.9. Design Information Elements Extraction – Estimator C (Project C)	111
Figure 5.10. Plot of the Learning Curve for IEM Algorithm.....	116
Figure 5.11. Partial Material Result List for Sheathing and Gypsum Board.....	117
Figure 6.1. Proposed IMP Algorithm Development Method.	121
Figure 6.2. Map Structure of a Sample Constructor and HashMap.....	123
Figure 6.3. Flowchart of the Developed Item Matching Algorithm.	123
Figure 6.4. Material Layers of a wall (Commercial Acoustics 2017).	125
Figure 6.5. Experimental Cost Estimating Results (partial) using the Proposed Method and Corresponding Algorithms.....	127

LIST OF ABBREVIATIONS

QTO	Quantity Take-off	IEM	Information Extraction and Matching Algorithm
MVD	Model View Definition	CSC	Construction Specifications Canada
IFC	Industry Foundation Classes	CPIc	The Construction Project Information Committee
BIM	Building Information Modeling	NBS	National Building Specification
WBS	Work Breakdown Structure	USD	United States Dollars
CSI	Construction Specifications Institute	CAD	Computer-Aided Design
LOD	Level of Development/Detail	AEC	Architecture, Engineering, and Construction (AEC)
D-READ	Data-Driven Reverse Engineering Algorithm Development	IIE	Interpreted Information Exchange (IIE)
NLP	Natural Language Processing	NBIMS	National BIM Standard
UML	Unified Modeling Language		
SfB	Samarbetskommitten for Byggnadsfragor		
BSAB	Byggandets Samordning AB		
AIA	American Institute of Architects		
NSF	National Science Foundation		
GSA	General Services Administration		
IE	Information Extraction		

PREFACE

This dissertation contains published and submitted journal articles by the author while based in the discipline of Construction Management Technology, School of Construction Management Technology of Purdue Polytechnic Institute, West Lafayette, Indiana. The research was financially supported by the National Science Foundation (NSF) under Grants No. 1745374 and No. 1937115. Any opinions, findings, conclusions, or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the NSF. The research was also financially supported by Dr. Jiansong Zhang's start-up fund.

This research's overarching goal was to develop and experimentally evaluate state-of-the-art techniques to improve the automation of cost estimation while also reducing human dependability issues in generating cost estimates. Six articles were developed from this work. Four of these articles are in print, and the other two are under review at the time of writing this dissertation.

Declaration of Published Works

This dissertation represents a compilation of manuscripts/ published work where each chapter were prepared as per the journals' specifications thus some repetition between chapters has been unavoidable. The first author (student) conducted all experimental work, data collection and manuscript preparation, under the guidance of the second and/or third author (advisor).

ABSTRACT

Cost estimation is an integral part of any project, and accuracy in the cost estimation process is critical in achieving a successful project. Manually computing cost estimates is mentally draining, difficult to compute, and error-prone. Manual cost estimate computation is a task that requires experience. The use of automated techniques can improve the accuracy of estimates and vastly improve the cost estimation process. Two main gaps in the automation of construction cost estimation are: (1) the lack of interoperability between different software platforms, and (2) the need for manual inputs to complete quantity take-off (QTO) and cost estimation. To address these gaps, this research proposed a new systems to support the computing of cost estimation using Model View Definition (MVD)-based checking, industry foundation classes (IFC) geometric analysis, logic-based reasoning, natural language processing (NLP), and automated 3D image generation to reduce/eliminate the labor-intensive, tedious, manual efforts needed in completing construction cost estimation. In this research, new IFC-based systems were developed: (1) Modeling – an automated IFC-based system for generating 3D information models from 2D PDF plans; (2) QTO - a construction MVD specification for IFC model checking to prepare for cost estimation analysis and a new algorithm development method that computes quantities using the geometric analysis of wooden building objects in an IFC-based building information modeling (BIM) and extracts the material variables needed for cost estimation through item matching based on natural language processing; and (3) Costing – an ontology-based cost model for extracting design information from construction specifications and using the extracted information to retrieve the pricing of the materials for a robust cost information provision.

These systems developed were tested on different projects. Compared with the industry's current practices, the developed systems were more robust in the automated processing of drawings, specifications, and IFC models to compute material quantities and generate cost estimates. Experimental results showed that: (1) Modeling - the developed component can be utilized in developing algorithms that can generate 3D models and IFC output files from Portable Document Format (PDF) bridge drawings in a semi-automated fashion. The developed algorithms utilized 3.33% of the time it took using the current state-of-the-art method to generate a 3D model, and the generated models were of comparative quality; (2) QTO – the results obtained using the developed component were consistent with the state-of-the-art commercial software. However, the

results generated using the proposed component were more robust about the different BIM authoring tools and workflows used; (3) Extraction – the algorithms developed in the extraction component achieved 99.2% precision and 99.2% recall (i.e., 99.2% F1-measure) for extracted design information instances; 100% precision and 96.5% recall (i.e., 98.2% F1-measure) for extracted materials from the database; and (4) Costing - the developed algorithms in the costing component successfully computed the cost estimates and reduced the need for manual input in matching building components with cost items.

CHAPTER 1. INTRODUCTION

1.1 Motivation and Overview

Construction cost estimation involves the process of calculating costs based on the design components of a structure. Accuracy in this process is intrinsic in accomplishing a successful project. To generate a cost estimate, an estimator needs in-depth understanding of the project requirements; that is, the scope of work required to achieve the project. These project requirements are usually defined in the drawings and specifications. In generating an accurate cost estimate in the design stage, the first step for an estimator is the visualization of the project. The second step is quantification, i.e., the generation of quantities. The last step would be costing, i.e., the application of unit costs to the quantities generated in the second step. The accuracy of the cost estimate is usually dependent on the experience, technical expertise and the accuracy of the tools utilized in generating the cost estimate. Currently, most cost estimation tools still heavily rely on manual processes in generating quantities and computing cost estimates, BIM-based processes can vastly improve both the accuracy and time efficiency involved in the cost estimation process. This dissertation aimed to develop an IFC/BIM-based framework that addresses BIM interoperability and the heavy reliance on manual efforts in the generation of design cost estimates for the built environment, with a focus on bridges and multifamily units.

1.2 Background

Currently, for various commercial estimating software in use in the construction industry, the processes involved in achieving the estimates are similar. Most software requires a three-step operation to achieve cost estimation (Figure 1.1):

Step 1: Generating the 3D model - the building model is designed and an estimator imports the building design in BIM (or 3D drawings).

Step 2: Assigning model elements to categories (Step 2a) and taking off the quantities (Step 2b) - the estimator classifies elements of the model into different categories (e.g., using classification systems such as MasterFormat), and the software executes the quantity computations to output a bill of quantities.

Step 3: Apply applicable ‘Unit Cost’ - the quantities generated are imported and linked to pricing items in a cost database to assign costs to the various categories from Step 2 and calculate the final cost estimates.

As an example, using Autodesk Navisworks: (1) Step 1 - the generated 3D model is imported. The Navisworks platform can directly open some proprietary BIM 3D extensions; however, to ensure a full information exchange across both platforms, it is recommended to export the 3D models as a Navisworks extension (*.nwc); (2) Step 2a - the classification system is chosen to build the work breakdown structure (WBS), the measurement units are chosen and elements are manually mapped to their WBS; (3) Step 2b – take off quantities and export the report generated; (4) Step 3 – apply applicable unit cost from historical data to the quantities generated from Step 2b to generate the cost estimate. Similar to Autodesk Navisworks, using Trimble Vico Estimator: (1) Step 1- the 3D model is imported; (2) Step 2a – assign model elements to categories; (3) Step 2b – take off quantities; (4) Step 3 – apply applicable cost by importing data from existing projects or exporting quantities generated for pricing calculation.

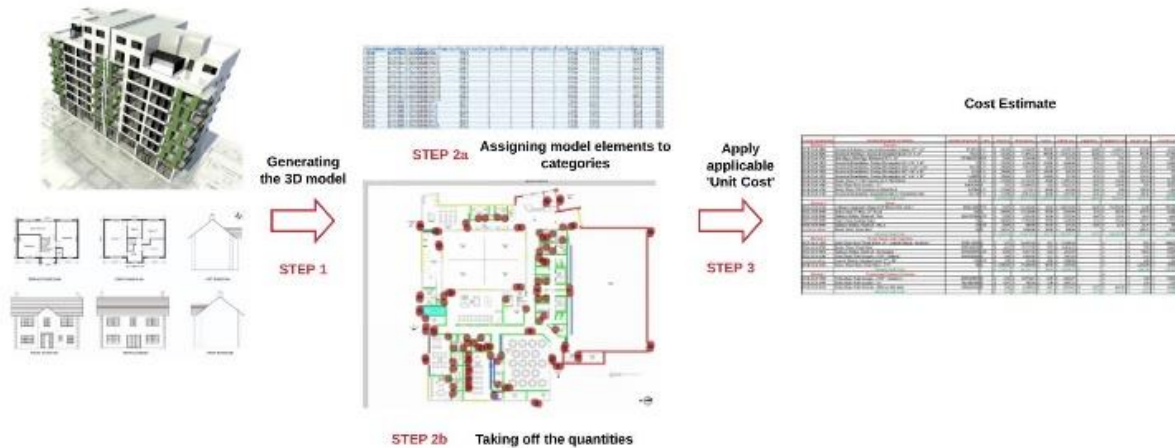


Figure 1.1. Typical Steps in Cost Estimation using Software

1.3 Significance

As shown in Figure 1.1, two main challenges posed by these current methods are the issues in collaboration (thus interoperability) and the needed human input. For the collaboration/interoperability issue, Step 1, and Step 3 of this general cost estimation process

(Figure 1.1) involve collaborative data exchanges. Different platforms store and keep information in different proprietary formats, impeding the successful exchange of data. Without a reliable, complete, and successful data exchange, the reliability of cost estimates is questionable. For the second issue in human input needs, although QTO has been automated with the advent of BIM, manual efforts by human estimators are still required in classifying and matching the model elements to their various categories (Lee et al. 2014). This can introduce human errors and therefore affect cost estimation accuracy. For example, to develop the cost estimate of a composite wall system with coding “CW 102-85-140p”, an estimator needs to classify the different components of the wall system appropriately, namely, face brick, air barrier, wall sheathing, insulation, wood frame, and gypsum board. This will be done based on the classification system employed. There are several established construction classification systems, such as Samarbetskommitten for Byggnadsfrågor (SfB) and Byggnadets Samordning AB (BSAB) in Sweden, Uniclass in the United Kingdom, Building 90 in Finland, MasterFormat and OmniClass in North America (Afsari & Eastman, 2016). Among these construction classification systems, MasterFormat, published by the Construction Specifications Institute (CSI), is one of the most commonly used classification systems in such capacity (Abanda et al. 2017). In the MasterFormat classification system, typically, a single composite wall system could be classified under three or four divisions, depending on the estimator. One estimator may select Division 4 (Masonry), Division 7 (Thermal and Moisture Protection), and Division 9 (Finishes), corresponding to face brick, insulation, and gypsum board assembly components, respectively. Another estimator may select Division 6 (Wood, Plastics, and Composites) in addition to the Divisions of 4, 7, and 9, to classify the same wall system, corresponding to the wood framing components. This difference in classification can result in significant differences in the cost estimation results. The estimator’s subjectivity and proficiency are further introduced when defining the activities to erect the wall or the finish of the wall as these activities/material choices differ based on an estimator’s knowledge, expertise, logic of reasoning and/or a company’s standard operating procedure. One estimator may consider two coats paint smooth finish (brushwork), whereas another may consider three coats paint smooth finish (sprayed); this information is not readily retrieved from BIM models and could affect the cost estimate.

To address these research gaps in BIM interoperability to support cost estimation, this research integrated 3D model generation methods, logic-based reasoning, and NLP processes into

a new IFC-based framework for automated construction cost estimation (Figure 1.2). As shown in Figure 1.2, the framework comprises of four components: (1) the modeling component – for the automated generation of 3D models from 2D PDF drawings and further conversion to IFC files which serves as input for the quantification component; (2) the quantification component – for the automated generation of QTO; (3) the extraction component – for the automated generation of design information from construction specifications; and (4) the costing component – for the automated generation of cost estimate computations leveraging the information extracted from both the quantification and extraction components.

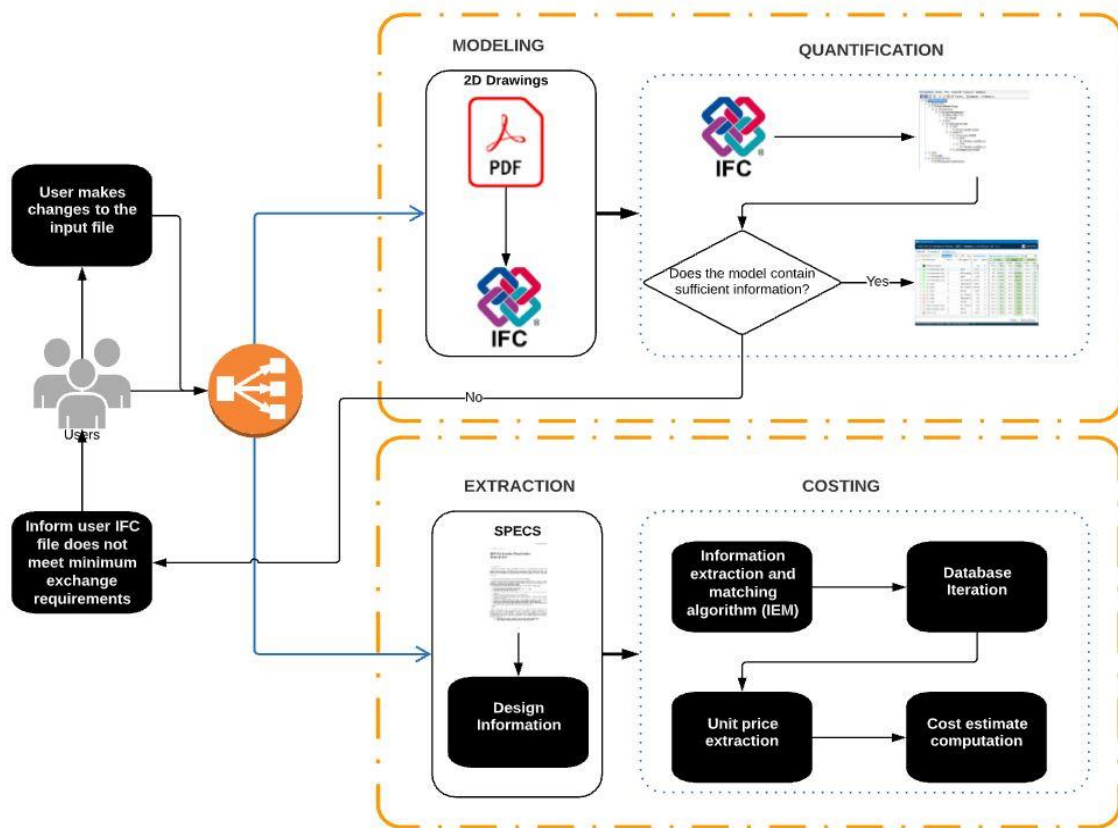


Figure 1.2. IFC-Based Framework to Support Cost Estimation

1.4 Problem Statement

This research aimed to address the gaps in: (1) Building Information Modeling (BIM) interoperability in construction to support cost estimation; and (2) reducing subjectivity and manual efforts in cost estimates' computations.

1.5 Research Questions

The questions fundamental to this research were:

1. What are the current state-of-the-art methods/techniques available for construction cost estimation?
2. What are the challenges posed by these identified methods/techniques?
3. Can an expert intelligent modeling system be developed that addresses the uncertainties of information exchange across different platforms required for construction cost estimation?
4. Can an expert quantification system be developed that reduces the need to classify and match model elements to their various categories manually?
5. Can an expert costing system be developed that can automatically retrieve missing BIM design information from construction specifications?

1.6 Assumptions

The proposed framework was based on the following assumptions:

1. The architectural models were in 3D BIM formats.
2. The 3D BIM can be converted to an IFC file – the system's input file.
3. The 3D BIM were at a Level of Development/Detail (LOD) 300 level or higher.
4. The construction specifications were following the MasterFormat classification system.
5. The drawing plans contained orthographic views of the project.

1.7 Limitations

Six main limitations are acknowledged in this research:

1. The developed 3D Model Generation Algorithm tracing map has a threshold that may omit small details of the structure.
2. The Data-Driven Reverse Engineering Algorithm Development (D-READ) method produced QTO algorithms that could only address geometric representations observed in the development data.
3. The QTO algorithms currently were tested on explicitly modeled elements (e.g., walls, slabs), unmodeled elements (e.g., scaffolding) were not tested.
4. The Information Extraction and Matching Algorithm only addressed design instances for wood elements observed in the development data.
5. The developed Information Extraction and Matching Algorithm were specifically tested on wood objects.
6. The developed Information Extraction and Matching Algorithm only addressed the MasterFormat construction specification classification system.

1.8 Delimitations

The following delimitations are identified as part of this research:

1. The algorithms for the modeling and quantification components were limited to specific elements of the built environment – bridges and multifamily units. These specific elements of the built environment were chosen because: (1) current BIM-based platforms for these elements are not fully developed to process traditional 2D drawings; and (2) in the United States, BIM tools are not fully utilized in developing models for single family residential structures. The QTO algorithm description, Information Extraction, and Matching Algorithm in the research focused on wood components. Wood was selected because: (a) wood is one of the major structural material that is renewable; and (b) wood structure accounts for most residential constructions in the United States.
2. The Information Extraction and Matching Algorithm description focused on the MasterFormat construction specification classification system only.

1.9 Research Aims and Objectives

This research aimed to investigate, assess, and develop automated IFC – based systems and/or methods to support construction cost estimation. To achieve these aims, a framework was developed that comprised of four components and the following objectives were pursued:

1. Research current intelligent methods/techniques of computing construction cost estimates to identify gaps.
2. Develop the modeling component of the framework – a method to generate IFC files from existing 2D drawings.
3. Develop the quantification component of the framework – a Data-Driven Reverse Engineering Algorithm Development (D-READ) method for interoperable quantity take-off using IFC-Based BIM.
4. Develop the extraction component of the framework – an automated Design Information Extraction method from construction specifications.
5. Develop the costing component - an automated Item Matching and Pricing method to generate cost estimates' computations.

1.10 Outline of the Dissertation Structure

This research comprises seven chapters presented in an article-based format. This chapter provides an overview of the research, including the background, significance, purpose, research questions, limitations, and delimitations. The overarching goal of this research was to develop a framework to support construction cost estimation. The developed framework comprised of four components – modeling, quantification, extraction, and costing. Chapters 3 to 6 are focused on presenting the automated IFC-based BIM methods developed in each of the four components of the framework. Each of these chapters contains the background section, the proposed methodology section, the experiment section, the experimental results section, the conclusions, contributions, limitations, and recommendations for future research section.

Chapters 2 describes the relevant literature in the following research areas: cost estimation, building information modeling, industry foundation classes, and natural language processing.

Chapter 3 investigates the use of generative modeling in generating 3D information models and IFC output files from traditional 2D plans. The proposed method in Chapter 3 exploits the

wealth of 3D shape generation and python in generating 3D models. The proposed modeling component of the framework developed in Chapter 3 was evaluated on developing algorithms for bridge structures.

Chapter 4 investigates the use of an IFC-based QTO method that supports BIM data from different BIM authoring tools/workflow. The developed quantification component of the framework in Chapter 4 can be utilized to develop QTO algorithms for any building component. The developed method can be applied to models developed using any IFC-compatible BIM platform.

Chapter 5 explores the use of logic-based information extraction methods for the automated extraction of cost information from construction specifications. The developed extraction component of the framework in Chapter 5 exploits the wealth of NLP techniques in extracting cost-related information and matching the extracted information to a cost database created using logic-based reasoning.

Chapter 6 explores the viability of a new automated method to reduce manual inputs needed from estimators in BIM-based cost estimation computations. The developed costing component of the framework in Chapter 6 computes cost estimates from a linked cost database, using an algorithm based on term-based match and natural language processing (NLP) techniques.

The final chapter, Chapter 7, provides a summary and conclusion of the framework of the complete construction cost estimation computation system, integrating the various components of research in Chapters 3 – 6. Chapter 7 further explores the various findings from the previous chapters and provides the research limitations and recommendations for future research sections.

CHAPTER 2. LITERATURE REVIEW

This chapter describes the reviewed literature in the following research areas, which are related to this dissertation. These research areas are cost estimation, building information modeling, industry foundation classes, and natural language processing. A version of this chapter has been published in the American Society of Civil Engineers (ASCE) Journal of Computing in Civil Engineering and proceedings from the ASCE Construction Research Congress.

Akanbi, T., Zhang, J., and Lee, Y-C. (2020). "Data-Driven Reverse Engineering Algorithm Development Method for Developing Interoperable Quantity Takeoff Algorithms Using IFC-Based BIM." *Journal of Computing in Civil Engineering*, 34(5): 04020036. DOI: 10.1061/(ASCE)CP.1943-5487.0000909. "With permission from ASCE"

Akanbi, T., Zhang, J. (2020). "Automated Design Information Extraction from Construction Specifications to Support Wood Construction Cost Estimation." *Proc., 2020 ASCE Construction Research Congress*, ASCE, Reston, VA, 658-666. DOI: 10.1061/9780784482889.069. "With permission from ASCE"

2.1 Cost Estimation

Cost estimation is critical to the success of a construction project (Yu et al. 2006; Choi et al. 2015). When conducting cost estimation, the main challenge for human estimators lies in the need of understanding the design conditions and selecting the appropriate quantities and cost parameters, which would have a deciding effect on the construction cost (Staub-French et al. 2003). Many items can influence a construction project cost; examples include engineering complexities, construction complexities, inflation, delivery/procurement approach, and bias (Shane et al. 2009). An estimator's bias (or preference) accounts for most of the disparities found in the cost estimates prepared by different estimators. This lack of uniformity (or inconsistencies) in the construction cost estimates from different estimators for the same building project results from the limitations of the manual processes in cost estimation (Staub-French et al. 2003). Manual cost estimation is a

tedious, time-consuming, and usually involves human errors (Samphaongoen 2009; Mandava and Zhang 2016).

2.1.1 Work Breakdown Structure (WBS) and Cost Breakdown Structure (CBS)

Work breakdown structures (WBS) and cost breakdown structures (CBS) are utilized in generating cost estimates (Polonski 2015). The WBS is an important tool/structure to construction estimators because it describes the details of the work required to accomplish a project (Jones et al. 2015). It decomposes a project into manageable sections of work to facilitate the planning and control of schedule and cost (Jones et al. 2015). The CBS on the other hand mirrors the WBS in the development of a cost estimate for all activities in completing the project; the CBS generates a cost for each element in the WBS (Jones et al. 2015). WBS is established using the various construction classification systems designed to catalog the built environment (Autodesk 2017). There are numerous established classification systems used around the world such as Samarbetskommitten for Byggnadsfrågor (SfB) and Byggnadets Samordning AB (BSAB) in Sweden, Uniclass in the United Kingdom, Building 90 in Finland, MasterFormat and OminClass in North America (Jorgensen 2011; Afsari and Eastman 2016). Among these construction classification systems, the most commonly used ones in the United States are MasterFormat, UniFormat, Uniclass, and OmniClass (Autodesk 2017). MasterFormat, UniFormat, and OmniClass are produced by the Construction Specifications Institute (CSI) and Construction Specifications Canada (CSC), while Uniclass was created in the United Kingdom by the Construction Project Information Committee (CPIc) and the National Building Specification (NBS) (Autodesk 2017). The General Services Administration (GSA) decrees that cost estimates for projects in the United States should be reported using the UniFormat II or MasterFormat classification systems (The General Service Administration 2017). Furthermore, UniFormat II elements classification system and MasterFormat classifications system are the recommended classification systems for the preparation of design specifications (Charette and Marshall 1999). In accordance with the “CSI/180 FF practice guide,” the CSI recommends the preparation of preliminary project description using the UniFormat II elements classification system and the use of the MasterFormat classification system for the preparation of outline and construction specifications (Charette and Marshall 1999).

2.1.1.1 MasterFormat Construction Specifications

MasterFormat maintains a master list of numbers and titles for cataloguing, classifying, and organizing construction work or activities (Autodesk 2017). In North America, MasterFormat is the accepted industry standard for building design and construction (Afsari and Eastman 2016). The MasterFormat classification system has 50 divisions (divisions 0 through 49) and is utilized for writing construction specifications and for enhancing communications across various project stakeholders (contractors, fabricators, specifiers, and suppliers) (CSI 2016). As shown in Figure 2.1, construction specifications are classified using a hierarchical structure consisting of four levels (levels 1 - 4). Project specifications are decomposed into divisions, which are further broken down into sections to provide details about the requirements of a specific product, material, or activity. Each section of the construction specification is further categorized into parts using the SectionFormat structure jointly produced by the CSI and CSC. SectionFormat helps provide a structure for construction specifications by grouping project requirements into three parts, namely: “Part 1 General,” “Part 2 Products,” and “Part 3 Execution” (CSI 1997). “Part 1 General” provides details about the administrative, procedural, and temporary requirements of the construction project (CSI 1997). “Part 2 Products” provides details about the materials, equipment, and product requirements of the construction project (CSI 1997). “Part 3 Execution” provides details of procedures to carry out the various tasks (CSI 1997). Each part within the construction specification contains one or more clauses, and each clause may contain one or more sub-clauses or paragraphs. A large percentage of the information required for cost estimation is specified in the product requirements group (i.e., “Part 2 Products”).

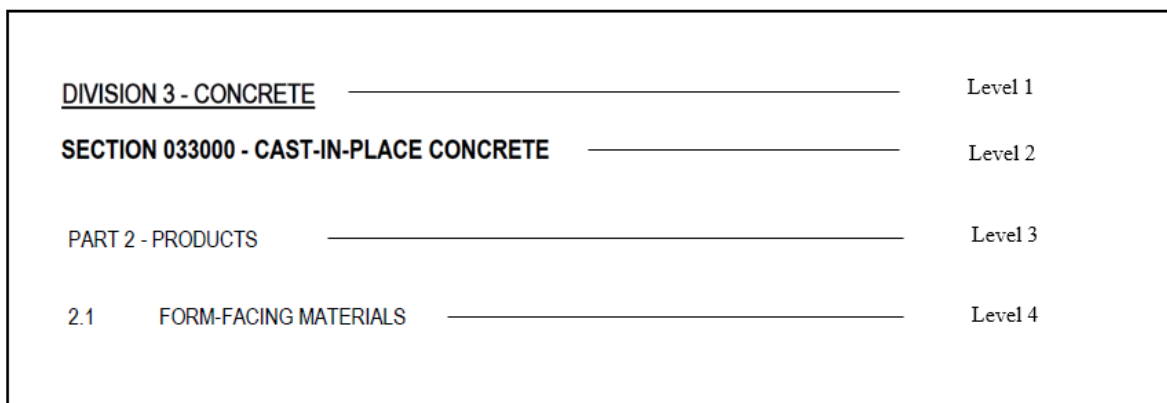


Figure 2.1. Example MasterFormat Structure of Construction Specifications

2.2 Building Information Modeling

Building Information Modeling (BIM) revolutionizes how architects, engineers, and construction contractors conduct their business (Franco et al. 2015; Lu et al. 2013; Nassar, 2012). BIM is defined to be “a digital representation of physical and functional characteristics of a facility. As such it serves as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its lifecycle from inception onward.” (The National Institute of Building Sciences 2007). It is a mechanism for producing building data using computer-aided design (CAD) and information technology tools (Aladag et al. 2016). BIM can be utilized to plan, construct, operate and maintain buildings and infrastructures (The National Institute of Building Sciences 2007). BIM’s adoption in the architecture, engineering, and construction (AEC) industry has been increasing. For example, from 2007 to 2012, the adoption of BIM in the AEC industry grew from 28% to 71% in North America (McGraw-Hill 2014). With the AEC industry experiencing this change from the traditional 2D concepts to the 3D BIM-based concepts, the General Services Administration (GSA) has mandated model-based designs and Industry Foundation Classes (IFC)-based BIM deliverables for all new Public Building Service projects (Matta 2006). BIM tools are comprehensive and are consisted of architectural design tools (e.g., Revit and ArchiCAD), structural design tools (e.g., Tekla and STAAD Pro), energy assessment tools (e.g., VE pro and Simuwatt), and cost estimation tools (e.g., DesignEst Pro and Autodesk Naviswork), among others (Cheung et al. 2012). Studies showed that BIM tools could benefit owners and construction professionals in various ways, such as design optimization (Bucarelli et al. 2018), schedule optimization (Zhang and Laddipeerla 2018), project coordination support (Ren et al. 2018; Ren and Zhang 2019,2020), and cost estimation automation (Mandava and Zhang 2018). When exploited, these benefits can help reduce project cost, increase productivity, and reduce rework or requests for information (RFIs) (Santos et al. 2017).

2.2.1 Level of Development/Detail

The level of development/detail (LOD) specification is a reference tool designed to enhance communications among BIM stakeholders regarding the components and elements in BIMs (BIMForum 2019). There are six basic LOD levels according to the AIA’s LOD schema:

(1) LOD 100 – models developed at the pre-design stage of a project;

- (2) LOD 200 – models developed at the schematic design stage of a project;
- (3) LOD 300 – models developed at the design development stage of a project;
- (4) LOD 350 – models developed during the construction documentation stage of a project;
- (5) LOD 400 – models documented during the construction stage of a project;
- (6) LOD 500 – models developed during the as-built documentation stage and close-out stage of a project.

The cost estimator extracts relevant information from a model to generate a cost estimate in the construction cost estimation domain. Information extracted from BIM at the different LODs is used to generate different types of estimates. Information extracted from BIM at LOD 300 & LOD 350 is used to generate rough cost estimate while information extracted from BIM at LOD 400 and higher are used to generate detailed cost estimate. In generating a detailed cost estimate for models with insufficient cost information (i.e., models at LOD 350 or lower), the estimator must link the extracted information from BIM to additional information to generate a cost estimate. This additional information is usually from construction specifications. According to Del Pico (2012), construction specifications define the detailed processes and materials required for the project and are issued during the construction phase. However, these manual processes of extracting cost information from construction specifications require in-depth construction knowledge/experience and are human error-prone (Ma et al. 2016).

2.2.2 Building Information Modeling in Vertical Construction

In the vertical construction domain, present BIM use was heavily focused on few purposes such as 3D coordination, design support, and clash detection (Kreider et al. 2010). BIM use in other functions needs further development to realize more potential benefits to building owners and industry professionals. One of the most useful such development is in automated quantity take-off (QTO) (Franco et al. 2015; Plebankiewicz et al. 2015; Monteiro and Martins 2013). As shown in the survey by Monteiro and Martins (2013), BIM-based QTO processes can vastly improve the accuracy and time efficiency of the cost estimation process. In as much as the study revealed, many efforts have utilized BIM tools to automate QTO. However, their survey results showed that automation of the complete estimating process (e.g., work item selection and costing) is still underexplored and requires further research/development. According to Nassar (2012), some designs are even still represented in 2D views. As such, at times, automated QTO from complete

3D models cannot be easily achieved. Currently, several methods, techniques, and software programs are available for cost estimation purposes. However, most commercial software programs such as Autodesk Navisworks and Vico Estimator use their proprietary data formats and still require estimators to manually match materials of building elements to work items (Lee et al. 2014). The use of BIM to support cost estimation is, therefore, not fully automated. In addition, the research interest in BIM-based cost estimation seems not as high as in other application areas. A BIM review on literature between 2005 and 2015 showed that only a few BIM studies focused on cost estimation (Santos et al. 2017). Even for the seemingly easy-to-achieve goal of automated QTO, the accuracy is not guaranteed, and it varies from method to method (Mittas et al. 2015).

2.2.3 Building Information Modeling in Horizontal Construction

In the infrastructure domain, various civil infrastructure projects have begun implementing BIM to primarily create an integrated 3D information model in aiding cost savings for all resources during the lifecycle of the infrastructure (Bae et al. 2016; Mastali and Zhang 2017). Kumar et al. (2017) listed several benefits of utilizing BIM on an infrastructure project, including: (1) enhanced coordination; (2) quick and improved clash detection; (3) enhanced productivity in design management; (4) improved risk management identification; and (5) efficient and accurate material quantities extraction. Similar to building projects, BIM tools have significant potential to add value across the life cycle of infrastructure projects (Fanning et al. 2014). In the preconstruction phase, BIM is used to assess project constructability and develop the project budget and schedule (Lau et al. 2018). In the construction phase, BIM enhances inter-trade coordination and provides quality control while the project is being executed. Through the lifecycle of the project, BIM is used in developing the maintenance plan. To take full advantage of the benefits obtained from the use of BIM in the design, construction, and maintenance of horizontal construction, current BIM development in the civil infrastructure domain is heavily focused on bridges (Kim et al. 2015; Cheng et al. 2016). Cheng et al. (2016) conducted an analytical review of academic efforts for developments of BIM-based methods/systems for infrastructures, the results revealed that out of sixty-two (62) academic papers reviewed, BIM-based methods for bridge construction had the most (27) papers whereas BIM-based methods for road construction (the second most) had just 8 papers. Although BIM offers these benefits for bridge design, construction, and maintenance; current BIM-based practices for bridges are not fully developed. Traditional 2D bridge drawings

and manual processes are still the de-facto standards. For example, currently, the inspections of bridges are mostly conducted manually (Isailovic et al. 2020). Trained professional engineers and practitioners perform periodic inspections to identify areas that require maintenance. Once these maintenance areas are identified, maintenance work items are then generated. Most of the processes in generating/computing the work items/quantities for these identified maintenance areas are usually computed manually or using systems that still rely heavily on manual inputs and approximate quantities. This is mainly because the existing bridge drawings are mostly in the traditional 2D format. In the industry, processing traditional 2D drawings (such as in quantity take-off) is typically conducted using on-screen take-off platforms. Despite improvement over using hardcopies, these processes still rely heavily on time-consuming manual efforts, cumbersome, and require years of bridge technical experience. To fully utilize the benefits of BIM for existing bridges, a 3D information model is required. The current industry practices are to develop these 3D information models from scratch using information manually retrieved from the record drawings.

2.3 Industry Foundation Classes

In achieving high accuracy, a successful data exchange among the different users of information is needed to avoid loss or misrepresentation of data (Nawari 2012). The foundation for BIM to accomplish smooth data exchange lies in its interoperability (Cheung et al. 2012; Santos et al. 2017). However, the lack of industry-wide interoperability has plagued the AEC industry since the embracement of BIM in the early 2000s (Eastman et al. 2011). This interoperability gap led to the alliance of twelve private companies to seek for full information exchange between the different software programs used in the AEC industry (buildingSMART). The alliance resulted in the establishment and development of industry foundation classes (IFC). The IFC standard is currently an ISO registered data standard for building and construction industry data – ISO 16739. It provides an open and neutral platform for information exchange within the AEC industry in a standardized way. However, despite establishing such standards, full interoperability (i.e., seamless information exchange) between different BIM software programs is yet to achieve. Gaps such as the lack of a proper exchange mechanism or software application that implements model-based interoperability still stand in the way (Sacks et al. 2010). Comprehensive interoperability solutions would require additional BIM research and development in many areas, such as (1)

widening the spectrum of project information; (2) developing a more detailed and generalized exchange standard; and (3) developing platforms that utilize model-based interoperability (Nawari 2012). Even for the widely accepted and ISO-registered IFC schema, further research and development is still needed. For example, Sacks et al. (2010) conducted an experiment to examine and analyze BIM and data exchange for precast concrete facades. Their results revealed that the IFC schema lacked the required property sets needed for successfully exchanging precast concrete models; there were also inconsistencies discovered in data exchanges between engineering and architectural systems. Seamless BIM interoperability is yet to be achieved. Similarly, Cheung et al. (2012) identified an interoperability problem using IFC-based BIMs, namely, the lack of conformity between the way IFC schemas are adopted by individual BIM tools, i.e., how IFC objects and properties are used. They proposed using best practices to guide data exchange between BIM tools to solve the current interoperability problem. Pauwels and Terkaj (2016) developed a converter that converts EXPRESS (i.e., the language used to write the IFC standard) schemas into web ontology language so that accurate and consistent distinctions could be made during data sharing. Wu and Zhang (2021) proposed a method that can be used to develop algorithms that automatically classify objects in an IFC model into predefined categories. Ren and Zhang (2021) developed a method to enhance data exchanges between architectural design and structural analysis. Wu et al. (2021) and Wu and Zhang (2018, 2019) developed invariant signatures to address the information irregularities during the conversion of models across BIM platforms. In the wood construction domain, Nawari (2012) reviewed the use of BIM in modeling wood structures and developed an information delivery manual (IDM) and model view definitions (MVDs). The IDM defines data indexes that must appear in the IFC schema for wood construction, whereas the MVD provides a protocol for how these data must be encoded in the schema. Such efforts potentially support the exchange of BIM data for wood construction in the future through IDM and MVD standardization, but there is no guarantee that these guides will be followed.

2.3.1 Model View Definition

The quality of IFC models varies. 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 among 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, the aggregated specifications of BIM data exchange requirements defined for a specific discipline, plays a pivotal role in providing 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 fulfill data exchange requirements, an MVD-based checking should be adopted to validate the accuracy of the IFC file (Lee et al. 2019). An MVD consists of a sequence of specification units referred to as ‘concept,’ which includes a 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.

2.4 Natural Language Processing (NLP)

Natural language processing (NLP) uses computational approaches to understand and produce human language content (Jurasky and Martin 2009; Hirschberg and Manning 2015). NLP involves approaches in artificial intelligence, computational linguistics, mathematics, and information science (Zhang et al. 2019). NLP techniques facilitate understanding, processing, and transformation of natural language text or speech using computers (Zhang and El-Gohary 2011; 2012a,b; 2013 a,b; 2014; 2015a,b,c; 2016; 2017). According to Crowston et al. (2011), NLP can be applied to provide automated analysis of textual data to detect and extract specific information from the data. NLP applications include text classification, language modeling, machine translation, and caption generation, among others (Zeng et al. 2015). Construction specifications are written textual documents that provide details about a construction project. Construction

specification document contains several types of information, ranging from quality assurance to design information for cost estimation. Each division in the MasterFormat CSI SectionFormat is organized into three parts. All the design information required for cost estimation is mostly contained in “Part 2 Products.” To extract this required information for cost estimation, NLP can be utilized to process the construction specifications, locate, extract the required pieces of information, and use the extracted information to match with materials in a database. Figures 2.2 and 2.3 represents partially an example specifications document, including “Parts 2 Products” of “Division 072100 – Thermal Insulation” and “Part 1, Section 1.2, Paragraph B” of “Section 092900 – Gypsum Board,” respectively. Figure 2.2 shows an example of identified entities to be extracted, while Figure 2.3 shows an example of the complex multi-layered and intertwined nature of construction specification documents. As depicted in both figures, developing algorithms that automatically extract information from construction specifications is challenging in two ways:

- (1) construction specifications contain different complicated text characteristics such as special characters, tabs, and whitespaces; and
- (2) construction specification sentences contain sub-sentences, and these sub-sentences may contain multiple design information pertaining to two or more building components or sub-components. In Figure 2.2, the spaces between texts, paragraphs, and non-alphabetic characters such as the quotation marks (“ ”) and semicolon (;) would need to be filtered out from the construction specifications to enhance the performance of extraction of the needed entities. Furthermore, construction specifications are semi-structured in nature. Hence, a text analysis would be required after preprocessing to extract all required cost design information. In Figure 2.3, “Section 092900 – Gypsum Board,” which contains information of gypsum boards, is related to “Section 061600 – Sheathing,” “Section 093013 – Ceramic Tiling,” and others. In developing the required information for each section, all referenced sections listed in the related requirements (e.g., Paragraph B of Part 2 – Products) for each section were cross-referenced. For example, for gypsum boards, sections pertaining to sheathing, gypsum board shaft wall assemblies, non-structural metal framing, gypsum veneer plastering, and ceramic plastering were cross-referenced. Once the required information to be extracted is established, the method utilizes NLP techniques in developing algorithms that extract

the required design information and automatically match the extracted design information with the unit price of materials from a database.

PART 2 - PRODUCTS

2.1 EXTRUDED POLYSTYRENE FOAM-PLASTIC BOARD

- A. Extruded polystyrene boards in this article are also called "XPS boards."
- B. Extruded Polystyrene Board, Type IV: ASTM C 578, Type IV, 25-psi minimum compressive strength; unfaced; maximum flame-spread and smoke-developed indexes of 25 and 450, respectively, per ASTM E 84.
 - 1. Manufacturers: Subject to compliance with requirements, available manufacturers offering products that may be incorporated into the Work include, but are not limited to the following:
 - a. DiversiFoam Products.
 - b. Dow Chemical Company (The).
 - c. Owens Corning.
 - 2. Fire Propagation Characteristics: Passes NFPA 285 testing as part of an approved assembly.

Figure 2.2. Partial Image of Part 2 – Products (Division 072100 – Thermal Insulation) of an Example Specifications Document

B. Related Requirements:



- | |
|--|
| 1. Section 061600 "Sheathing" for gypsum sheathing for exterior walls. |
|--|

GYPSUM BOARD

092900 - 1

09/14



- | |
|---|
| 2. Section 092116.23 "Gypsum Board Shaft Wall Assemblies" for metal shaft-wall framing, gypsum shaft liners, and other components of shaft-wall assemblies. |
|---|



- | |
|--|
| 3. Section 092216 "Non-Structural Metal Framing" for non-structural steel framing and suspension systems that support gypsum board panels. |
|--|



- | |
|---|
| 4. Section 092613 "Gypsum Veneer Plastering" for gypsum base for veneer plaster and for other components of gypsum-veneer-plaster finishes. |
|---|



- | |
|--|
| 5. Section 093013 "Ceramic Tiling" for cementitious backer units installed as substrates for ceramic tile. |
|--|

Figure 2.3. Partial Image of Section 092900 – Gypsum Board (Part 1, Section 1.2, Paragraphs B) of an Example Specifications Document

CHAPTER 3. MODELING COMPONENT - FRAMEWORK FOR DEVELOPING IFC-BASED 3D DOCUMENTATION FROM 2D DRAWINGS

A version of this chapter has been submitted to the ASCE Journal of Computing in Civil Engineering:

Akanbi, T., and Zhang, J. (2021). “Framework for Developing IFC-Based 3D Documentation from 2D Bridge Drawings.” *Journal of Computing in Civil Engineering*, submitted.

The chapter presents an approach for semi-automatically generating a 3D information model from 2D PDF drawings in the industry foundation classes (IFC) format. As a result of the interoperability gap in the AEC industry, a non-proprietary standard for uniform information exchanges is the most promising direction. IFC is a registered non-proprietary standard for uniform information exchange within the AEC industry. IFC is a comprehensive, robust data model that is supporting software vendor-independent BIM data exchange (Huthwohl et al. 2018). The proposed component was evaluated by utilizing the proposed component in generating algorithms that can automatically: (1) process existing 2D bridge drawings for bridges-built pre-BIM adoption in the architecture, engineering, and construction (AEC) industry; (2) convert these record drawings to 3D information models; (3) convert the 3D information models to industry foundation classes (IFC) files. Bridges were selected because although BIM-based platforms can provide many benefits to Department of Transportations (DOTs), current BIM-based platforms for bridges are not fully developed to process traditional 2D bridge drawings for BIM-based computational tasks of existing bridges, e.g., BIM-based cost estimation.

3.1 3D Shape Generation

A few researchers have proposed methods and systems for geometric shape representation of objects using generative modeling (Lin et al. 2017). “Generative modeling is a process that involves automatically discovering and learning the regularities or patterns in an input data in such a way that the model can be used to generate outputs that plausibly could have been drawn from the original dataset” (Brownlee 2019). According to Girdhar et al. (2016), an object representation must satisfy two benchmarks: (1) generative 3D – a user should be able to generate a 3D

representation from the object representation; (2) 2D predictability – a user should be able to construe the object representation from images. In this paper, the author proposed constructing a three-dimensional representation of a bridge using vector graphics orthographic views of the bridge (Figure 3.1). An object (e.g., a bridge structure) can be represented by a series of univocal set of projections as shown in Figure 3.2, such as front projection (A), top projection (B), and side projection (C). A representation of a 3D wireframe model can then be generated utilizing these projections to form an object, a bridge structure in this case. There are typically four processes involved in generating such models from 2D images: (1) labelling of vertices; (2) topological representation of edges; (3) creation of intermediate vertices and collinear edges; and (4) creation of vertices and edges in 3D space (Furferi et al. 2010).

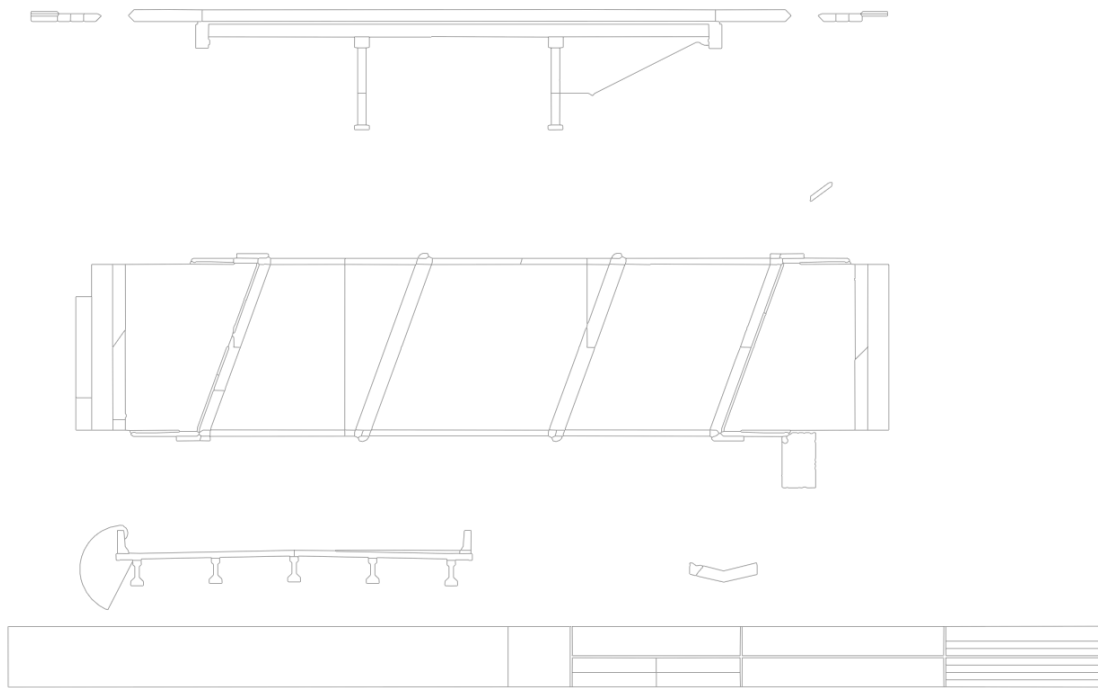


Figure 3.1. Orthographic Views of a Bridge

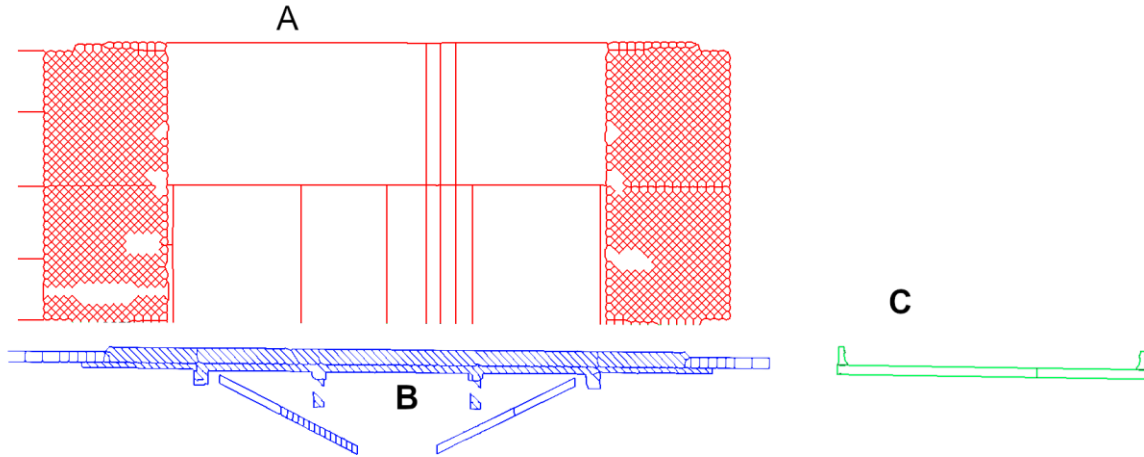


Figure 3.2. Projections of a Bridge Structure

3.2 Proposed Component for Generating 3D Information Model

To address the research gap in developing 3D information models of existing bridge structures, this dissertation proposed a component for the semi-automated generation of 3D information models and IFC files from 2D orthographic bridge drawings. The component can be used to develop algorithms that can: (1) process existing 2D bridge drawings; (2) convert these record drawings to 3D information models; and (3) further convert the 3D information models to IFC files. The component includes seven steps (Figure 3.3). Step 1: PDF Importation, this step imports the source file (PDF) of input bridge drawings; Step 2: PDF File Conversion, this step converts the imported 2D PDF file to a raster graphics format; Step 3: Raster Graphics File Selection, this step selects the required sheet(s) from the generated raster graphics file; Step 4: Raster Graphics File Cleaning and Conversion, this steps removes unnecessary texts from the selected sheet(s) from Step 3 and generates a vector graphics file; Step 5: Vector Graphics File Conversion, this step extracts the required main projections from the vector graphics file and converts the vector graphics file into a tagged data graphics file; Step 6: Tagged Data Graphics File Conversion, this step connects the cartesian points in the tagged data file and generates the 3D image object; Step 7: IFC File Generation, this step utilizes the ISO IFC standard to support the conversion of the 3D information model generated from the traditional 2D drawings, to IFC output files. The details of these seven steps are described in the following subsections, respectively.

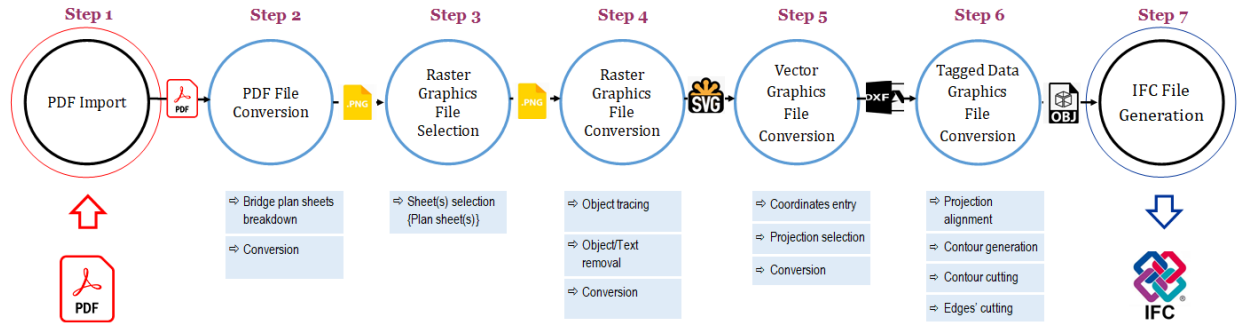


Figure 3.3. Proposed Component for Generating 3D Information Model

3.2.1 Step 1- PDF Import

Typically, the 2D bridge drawings are in a batch of PDF files. In this step, the bridge plans (batch of PDF files) are imported into the proposed system. There are different types of bridge plans produced at different phases of the design project – schematic design plans, design development plans, bidding design plans, the construction plans and the as-built plans. In this research the focus was limited to the bidding design plans.

3.2.2 Step 2- PDF File Conversion

The imported bridge plans from Step 1 are converted to a raster graphics file format. A raster image uses dot matrix structure composed of several fixed rectangular grids of pixels that make up a complete image (Australian National University 2020). There are a couple of raster format file types such as Joint Photographic Experts Group (JPEG), Graphics Interchange Format (GIF), Portable Network Graphics (PNG), etc. The PDF bridge plans contain different information such as the plan view projections of the bridge, cross-sections of the bridge, elevations of the bridge, and other information. In the further processing required for the generation of the 3D model, the PDF files need to be converted to a raster graphics format.

3.2.3 Step 3- Raster Graphics Sheets Selection

Once the PDF files have been processed and the raster graphics files are generated, the sheet(s) containing the required projections are selected from the batch of raster sheets created. The bridge PDF file contains several sheets such as the title sheet, index sheet, cross-section sheet, profile sheet, plan sheet(s), etc. The required projections are usually contained in the plan sheet(s). A plan sheet(s) would typically contain the plan of the bridge, the elevation of the bridge, and the typical cross-section of the bridge. However, the naming convention of the sheet(s) may vary slightly depending on the designer, or the intended user(s) of the bridge plans, such as the various owners, contractors, and consultants. In such scenarios, the user would have to manually select the sheet that contains the plan of the bridge, the elevation of the bridge, and the typical cross-section of the bridge.

3.2.4 Step 4- Raster Graphics File Conversion

The selected raster graphics' sheet(s) from Step 3, such as the general plan sheet(s), contains the required projections for the generation of the 3D model and other information irrelevant to our 3D information model generation process; examples of irrelevant information include letterings, plan titles, dimensions, etc. To expunge this irrelevant information, the raster graphics file is converted to a vector graphics file. Vector graphics images allow for more flexibility (scaling of shapes and changing colors) over raster graphics images and are defined in terms of points on a cartesian plane (MODassic 2020). The process of converting the raster graphics files to vector graphics file involves two main sub-steps (Figure 3.4): (a) Object tracing; and (b) Object/text removal. The details of these processes are described in the following sub-sections:

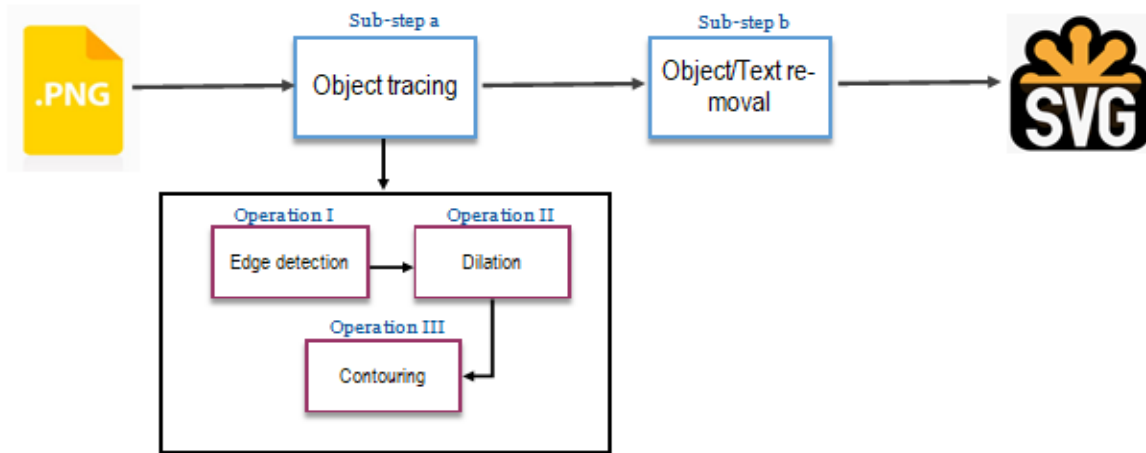


Figure 3.4. Processes Involved in Converting the Raster Graphics File to a Vector Graphics File

3.2.4.1 Sub-step a- Object tracing

A series of three operations are involved in the object tracing sub-step (Figure 3.4): Operation I - Edge detection, Operation II - Dilation, and Operation III - Contouring. Edge detection identifies the boundaries of an object within an image. Edge detection is necessary for the image segmentation and the extraction of required data from the selected raster graphics sheet(s). Once the object edges within the image are detected, the image is further processed by a morphological operation – Dilation. Dilation operation is used to emphasize the features of the objects and to join together detached parts of the objects within the image. The last operation under object tracing is contouring, which appends continuous lines along the object boundaries.

3.2.4.2 Sub-step b- Object/Text removal

In this sub-step, shapes with values that do not meet the pre-defined threshold are removed from the file. The threshold is defined as the minimum pixel value an object must attain to be included by our object tracing algorithm. A pre-defined value is selected based on the object to be processed. Any object with an intensity value lower than the threshold would be discarded and removed.

3.2.5 Step 5- Vector Graphics File Conversion

The vector graphics files generated from Step 4 are converted into tagged data graphics files. Tagged data graphics is a format used by computer-aided design (CAD) to produce 3D drawings. A tagged data graphics file is required to describe the object to be modelled mathematically. One tagged data graphics file is created for each projection. The process of converting vector graphics format to tagged data graphics format consists of two operations: coordinate entry and projection selection. The first operation inputs the coordinate entries of the selected projections, whereas the second operation selects and separates each projection. Similar to some other steps in the proposed component, the inputting of coordinate entries is conducted manually. A summary of all manual effort required using the proposed component is described at the end of the experiment section.

3.2.6 Step 6- Tagged Data File Conversion

This step is designed to be one of the main steps in processing and generating the 3D image projection. In this step, the lines between the cartesian points are created and utilized to generate the 3D images. The 3D model is displayed in fragments for batch processing to reduce the computing load and increase the 3D image processing speed. In processing the 3D models, a series of five sub-steps are used (Figure 3.5): (1) Align projections; (2) Create orthogonal lines; (3) Match coordinates; (4) Remove clones; and (5) Define intersection points. The details of these sub-processes are described in the following sub-sections:

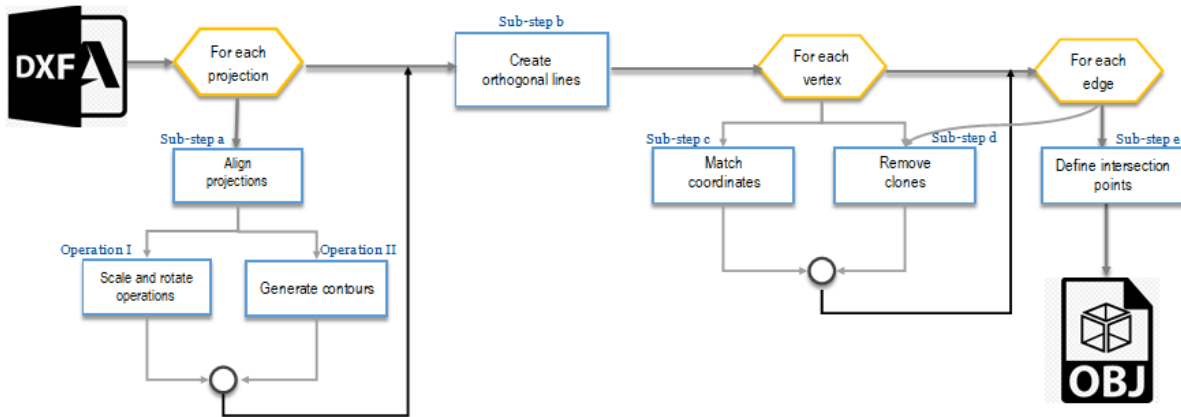


Figure 3.5. Processes Involved in Converting the Tagged Data Graphics File Format to OBJ File Format

3.2.6.1 Sub-step a- Aligning Projections

The projections from Step 5 (Vector Graphics File Conversion) are sometimes not aligned due to unmarked, hidden edges or edges that are not provided on the orthographic drawings. Figure 3.6a shows an example of the unaligned projections for a bridge, whereas Figure. 3.6b shows the aligned projection for the same bridge. In this sub-step, two operations are utilized to align the extracted projections: (1) Operation I – Scale and rotate operations; and (2) Operation II – Generate contours. The scale and rotate operations ensure that the virtual faces of each extracted projection are shifted to avoid errors between the extracted projections.

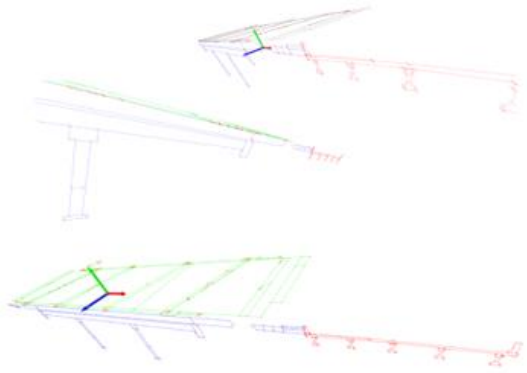


Figure 3.6a. Unaligned Projections of a bridge

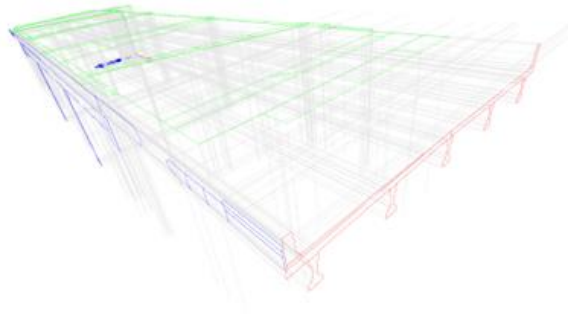


Figure 3.6b. Aligned Projections of a bridge

Figure 3.6. (a) Unaligned Projections of a Bridge; (b) Aligned Projections of a Bridge

3.2.6.2 Sub-step b- Creating Orthogonal Lines

This sub-step generates non-float rounded values for coordinates by creating orthogonal lines from projections to their opposite planes (Figure 3.7). As depicted in Figure 3.7, orthogonal lines (indexes) are created based on the vertices and corresponding edges generated in sub-step c. The generated contours from operation II (Sub-step a) produces both the needed contours for our 3D image generation and some unnecessary contours for the 3D image generation. The unnecessary contours generated are cropped and removed before the execution of this Sub-step b.

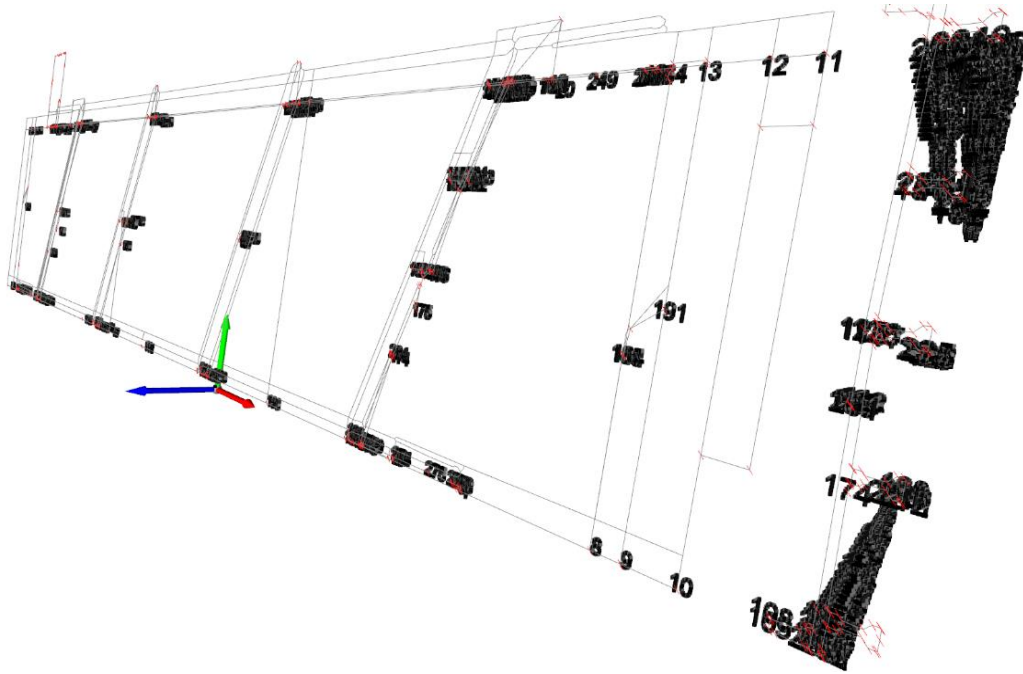


Figure 3.7. Sub-step b: Creating Orthogonal Lines

3.2.6.3 Sub-step c- Match Coordinates

In this sub-step, the arrays are optimized. By optimizing arrays, the coordinates are further matched by mapping the edges and vertices, and distortion in the structure to be generated is avoided. For vertices' coordinates to be matched, the distance between the vertices must be within an acceptable tolerance to indicate that the vertices are at the same point and can be matched. Figure 3.8 shows an example of matching coordinates. The black dots in the figure are the unique IDs of the vertices and edges, while the red lines show the unmatched coordinates. The matching of coordinates generates the mapping by comparing the unique IDs with concurrent vertices and edges.

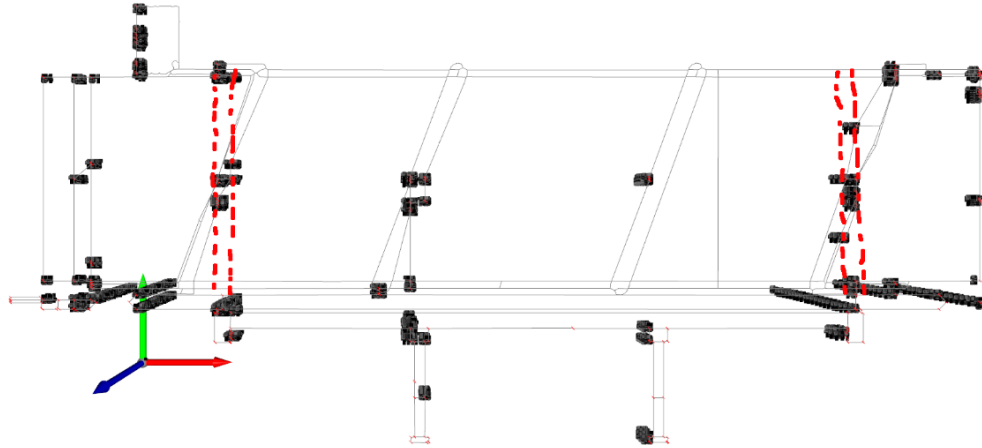


Figure 3.8. Sub-step c: Match Coordinates (Unmatched Coordinates Highlighted in Red)

3.2.6.4 Sub-step d- Remove Clones

For vertices and edges, clones are removed. First, the edges and vertices' intersection points are determined in the resulting model from sub-step c (match coordinates), and these new edges and vertices are added to the edge and vertices matrix. Next, object edges and vertices are verified by comparing the corresponding projection to each coordinate plane, model edge, or vertex. Each *uncertain* object edge or vertex contained in at least two noncoplanar virtual faces that do not belong to the projection can be identified and deleted. The process of deleting impossible virtual faces that do not meet the criterion and updating the virtual faces is an iterative process until a stable condition is achieved, i.e., the vertices and edges have unique matrixes.

3.2.6.5 Sub-step e- Define Intersection Points

In this sub-step, virtual cutting edges are introduced along the lines of intersections of the virtual faces. Furthermore, a list of siblings with common parent edges/faces and a list of correlations between the edges/faces that cannot co-exist in an object are generated. These data structures are used in this final stage of the 3D image development algorithms, where small independent virtual faces are removed. This sub-step is designed to be the last operation in processing and generating the 3D image projection.

3.2.7 Step 7- IFC File Generation

There are two outputs from Step 6 (Tagged Data File Conversion): (1) OBJ file – which contains the 3D mesh object; and (2) Encrypted bin file that contains the project data. The encrypted bin file contains the cartesian points and indexed polygonal face data of the projections essential in generating the IFC output file. Design data such as the geometric data contained in the bin file is complicated, and error usually arises when exchanging between different software platforms. In preventing such errors, this dissertation ensured the binary data are read and converted to standard codecs first. In this step, the data from the bin files are read, and the values are appended to their corresponding representation in the IFC file format.

3.3 Experimental Testing and Evaluation

To test the efficacy of the proposed component, the component was experimentally evaluated in generating algorithms for automatically developing 3D bridge models from 2D PDF drawings. The experiments were designed to test: (1) the accuracy of the 3D models generated; and (2) the time/effort saved by automating the processes involved in generating the 3D models using the proposed component.

3.3.1 Experimental Data

Eight bridges structures (Bridges A – H) were collected and used in the experiment. Bridges A and B were used to train and develop the 3D generation algorithms, while Bridges C – H were used in evaluating the accuracy and robustness of the developed algorithms. The bridge structures used in the evaluation were located in Indiana, USA. All bridge structures are continuous reinforced concrete slab bridges with over three spans, each span measured between 21' – 35'. Figure 3.9 shows the rendering of the bridges in Autodesk Revit 2021 (Autodesk 2021).

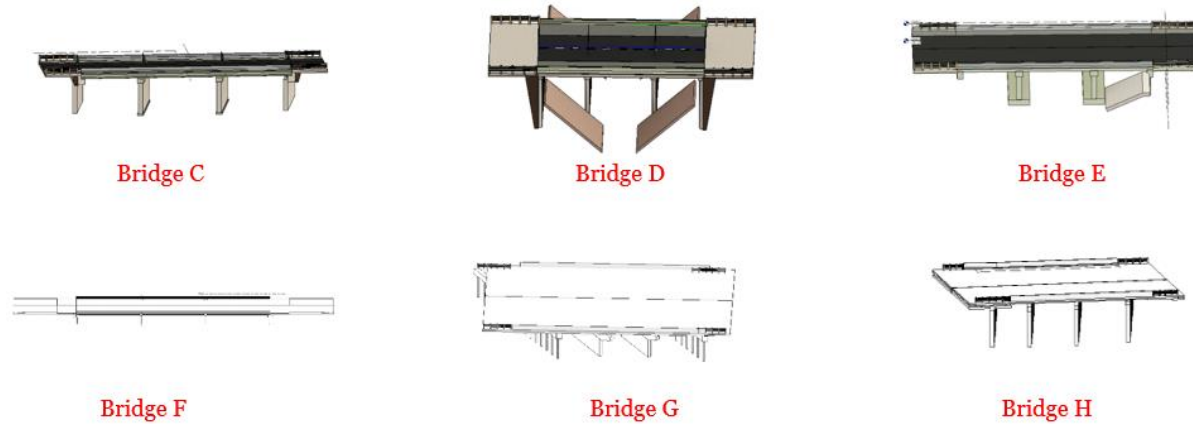


Figure 3.9. BIM Renderings of the Bridges

3.3.2 Evaluation

The developed algorithms from the proposed component were implemented in Python 3.9.0 (Python 2020) and utilized to process the 2D bridge plans. The results were compared with those obtained using current state-of-the-art practices in the industry. In analyzing the accuracy of the developed algorithms, the distance measurement between two cloud points was used to detect the model change and volumetric differences between two developed models (one using the developed algorithms and the other using the current industry-wide 3D bridge generation method). For each point cloud in the 3D model developed using Revit, CloudCompare searches the nearest point in the model developed using the algorithms and computes their Euclidean distances. The mean distance, Gaussian mean, and standard deviation were evaluated using the opensource 3D point cloud and mesh processing software – CloudCompare V2 (CloudCompare 2020). A partial result of the measurements of deviations (for Bridges C and D) are recorded in Table 3.2 as each 3D model generated has multiple points. As an example, the comparison for Bridge C produced 649 classes (i.e., cloud to cloud comparisons) results. A class is a set of two compared cloud points (one cloud point in each model).

3.3.3 Experiment

3.3.3.1 3D Model (Using Autodesk Revit)

The bridge plans for Bridges C – H were utilized to generate 3D models using Revit. Currently in the industry, professionals generate 3D bridge information models manually by

utilizing several BIM platforms such as Autodesk Revit and Bentley OpenBridge Modeler. The information used to generate these 3D models are usually extracted manually from 2D traditional bridge plans. In this experiment, the 3D models were developed by two industry experts (Designer I & II). Each model developed contained the geometric representations of the bridge extracted manually by each designer. Each designer was asked to individually: (1) read the blueprints/architectural bridge plans and manually extract the information required to generate the 3D bridge models; and (2) use Revit to generate the 3D bridge models. Each designer used Revit to model the site, the bridge structure, the topography and recorded the time it took to complete each bridge model from start to finish.

3.3.3.2 3D Model (Using Proposed Component)

The proposed component was used in developing algorithms that can process bridge PDF plans. These developed algorithms were then applied to processing the plans for Bridges C, D, E, F, G, and H and generate their respective 3D models.

3.3.3.2.1 Algorithms' Generation

Following the structure of the component, the algorithms generated consist of seven main steps; (1) Step 1 – Input; Step 2 – PDF File Conversion; Step 3 – Raster Graphics File Selection; Step 4 – Raster Graphics File Conversion; Step 5 -Vector Graphics File Conversion; Step 6 – Tagged Data Graphics File Conversion; and Step 7 – IFC File Generation. These seven steps further comprise twenty-five (25) processes and a decision; Figure 3.10 shows the partial flowchart of the algorithms developed to process the bridge plans. In developing the algorithms, several tools were utilized, including (1) an Optical Character Recognition (OCR) – ghostscript 9.53.3 in converting the PDF files to raster graphics format; an OCR software is used to analyze scanned PDF documents. Ghostscript served as an OCR tool to convert documents to printable formats (Artifex 2020); (2) the PNG raster graphics file format for the converted PDF files. PNG raster graphics file was used because it provided a well-compressed raster file for the system; (3) the scalable vector graphics (SVG) based in XML for the vector graphics file because the SVG was easily integrated with other specifications and standards; (4) the “opencv” library in python to convert the raster graphics sheet(s) to red-green-blue-alpha (RGBA) color models. An RGBA

3.3.3.2.3 Step 2- PDF File Conversion

After importing the architectural bridge plans, each bridge plan was converted to a raster graphics file. Figure 3.11 displays the algorithm for converting the PDF file to a raster graphics file. *Process I* loads all the sheets of the bridge plans in the PDF file. *Process II* iterates over the loaded pages to parse the data using the OCR software. In *Process III*, the OCR software is used to convert the pages to the raster graphics file format and generate one raster graphics file per sheet. *Process IV* saves the converted file, that is, the multiple raster graphics file sheets generated.

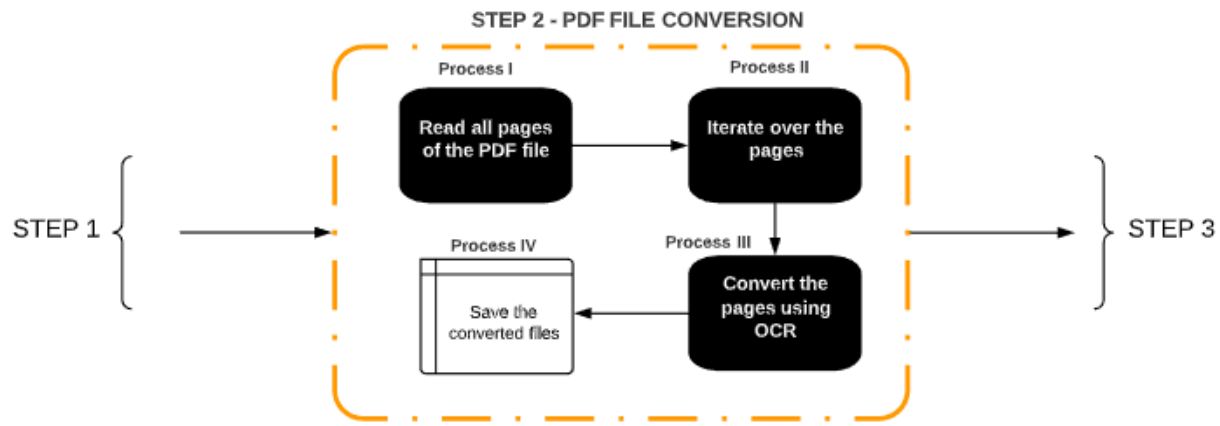


Figure 3.11. Step 2 Algorithm, PDF File Conversion

3.3.3.2.4 Step 3- Raster Graphics File Selection

Once the raster graphics files for all sheets are generated and saved, the required sheet(s) for the 3D model generation are selected from the batch of raster graphic sheets generated. Figure 3.12 displays the algorithm for selecting the required raster graphics sheet(s) from the saved file. *Decision I* checks if the saved file contains the sheet(s) for the required projections; if so, the algorithm proceeds to *Process V*, which then selects the sheet(s) with the required projections. If the file does not contain the required projections, the algorithm will prompt the user to re-import the PDF file with the required projections through Step 1.

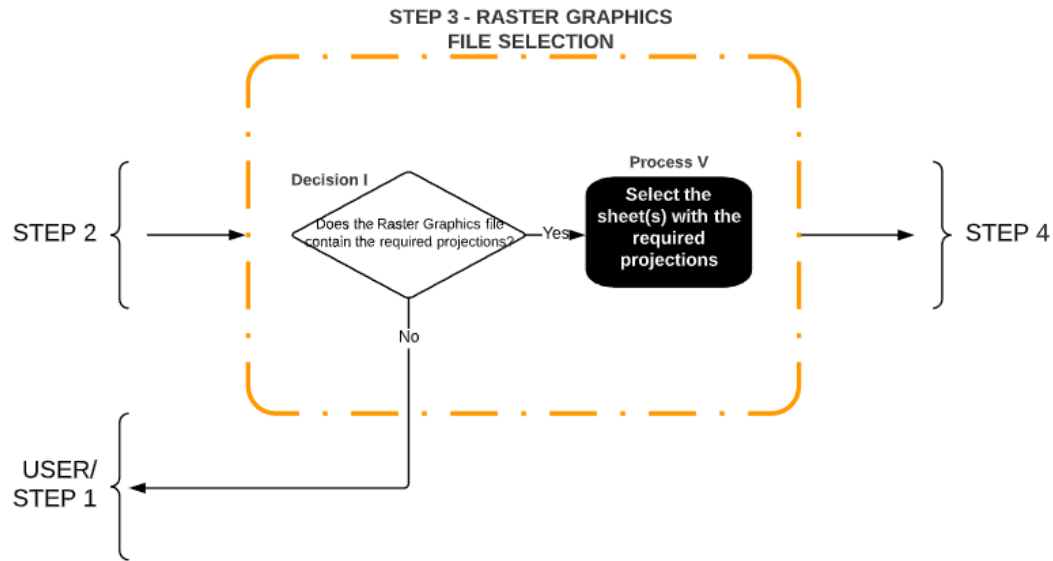


Figure 3.12. Step 3 Algorithm, Raster Graphics File Selection

3.3.3.2.5 Step 4- Raster Graphics File Conversion

The selected raster graphics sheet(s) from Step 3 typically contains other information irrelevant to our 3D information model generation process. Figure 3.13 displays the algorithm for converting the raster graphics file to a vector graphics file. *Process VI* utilizes the “opencv” library to convert the raster graphics sheet(s) to a red-green-blue-alpha (RGBA) color model. *Process VII* applied filters to uniquely identify the features and characteristics of the projections. In *Process VIII*, a gray-scaled mask is created. This process is required to mask the unnecessary features in the sheet(s). *Process IX* generates labels and slices required to complete the object tracing sub-step. The tracing map is configured to ignore minor, unimportant details in an event to prevent clogging of the final product with texts and minor details. In this experiment, we had set the minimum threshold to 500; that is, objects that are less than “50*10” pixels would be ignored. A lower threshold would accommodate more details but may also distort the model. *Process X* creates contours necessary to expunge the irrelevant texts/objects and exports them to the vector graphics file format utilizing the vector graphics creation standard in python. *Process XI* saves the exported file.

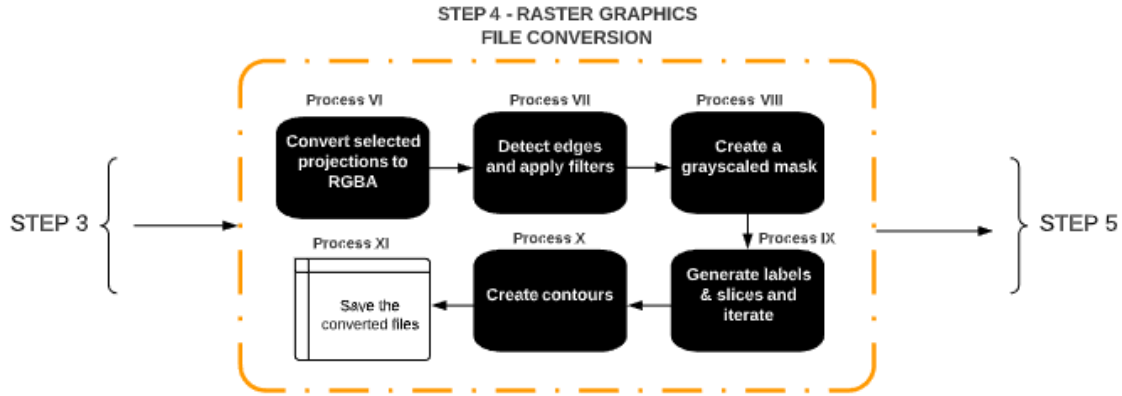


Figure 3.13. Step 4 Algorithm, Raster Graphics File Conversion

3.3.3.2.6 Step 5- Vector Graphics File Conversion

This step converts the vector graphics file generated in Step 4 to tagged data graphics file format. Figure 3.14 displays the algorithm for converting the vector graphics file to a tagged data graphics file. *Process XII* reads the saved vector graphics files in a vector graphics editor. The vector graphics editor software allows the retrieval of the boundary coordinates of each projection. *Processes XIII, XIV, and XV* are the main operations needed for the conversions of each projection (Figure 3.15). In Figure 3.15, letterings 1, 2, 3 represent the top, front, and side projections in Inkscape, while lettering 4 represents the coordinate entry (X, Y) of the minimum boundary point of the top projection. *Process XIII* retrieves the minimum entry coordinate for each projection, while *Process XIV* retrieves the maximum entry coordinates for each projection. *Process XV* enters the retrieved maximum and minimum entry coordinates in the system. The algorithm then proceeds to *Process XVI* to separate the projections into different files and convert the separated files into DXF file format using the “dxfwrite” module in python. This module allows for the exportation to the DXF file format. *Process XVII* saves the exported file format. Figure 3.16 shows an example of extracted bridge projections in a DXF file. A, C & E are the side projections; B is the top projection, and D is the front projection.

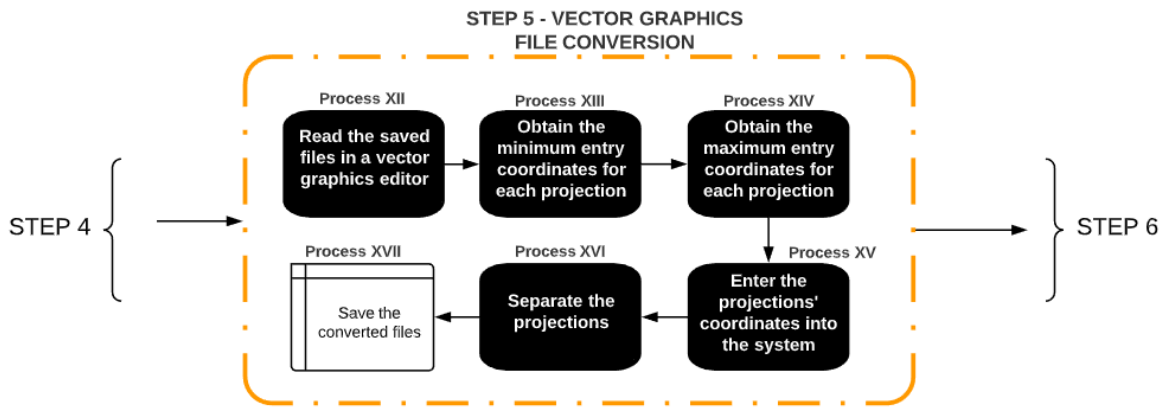


Figure 3.14. Step 5 Algorithm, Vector Graphics File Conversion

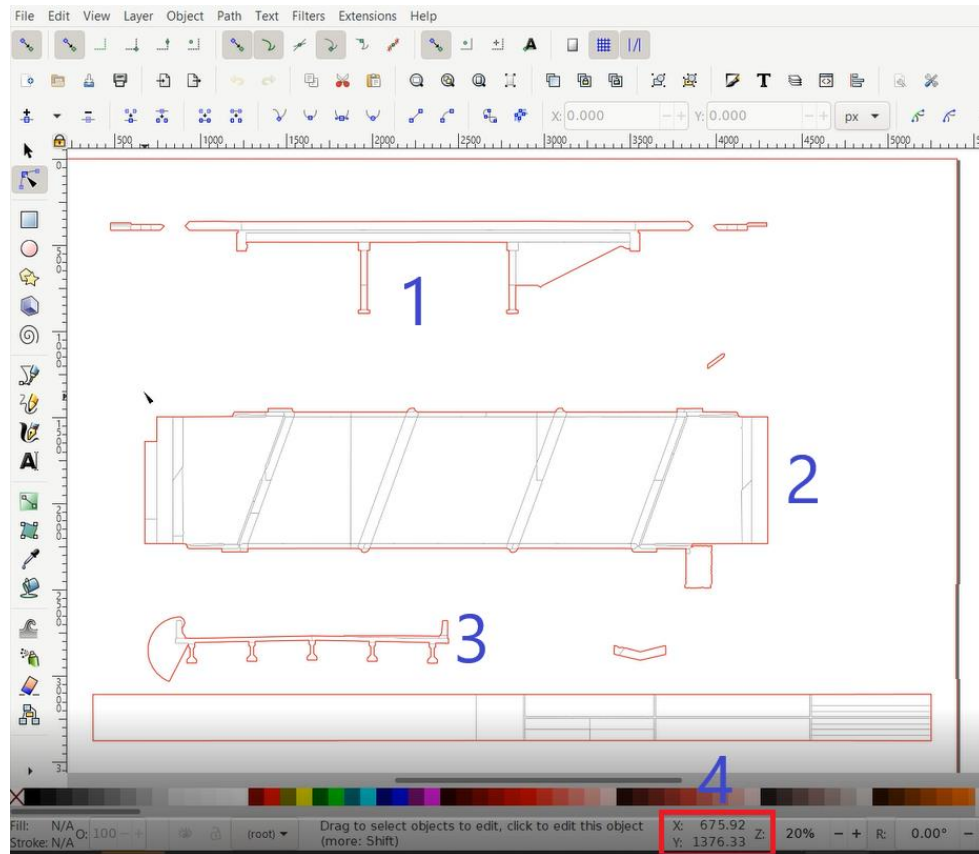


Figure 3.15. Projections of a Bridge Structure in a Scalable Vector Graphics File Format (Inkscape Graphics Editor)

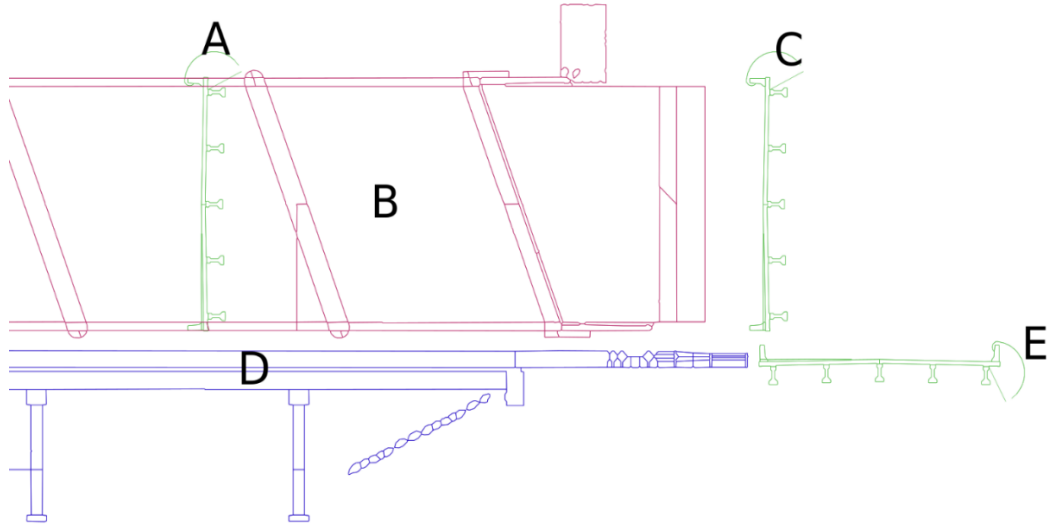


Figure 3.16. Projections of a Bridge Structure in DXF File Format

3.3.3.2.7 Step 6- Tagged Data Graphics File Conversion

This is the last main step in generating the 3D mesh model. Figure 3.17 displays the algorithm for converting the tagged data graphics file and generating the OBJ file and bin file. *Process XVIII* creates a list of vertices and triangles. In creating the list of vertices and triangles, the projections must be aligned by rotating the projections along the intersection of the matching lines. *Process XIX* generates the mapping of the vertices and triangles by iterating over the list of vertices and triangles. *Process XX* iterates over the mapped data, removes clones, creates the contours, and generates the 3D mesh object. *Process XXI* saves the generated 3D mesh object (Figure 3.18). Figure 3.18 displays an example of the 3D models generated for Bridges C and E.

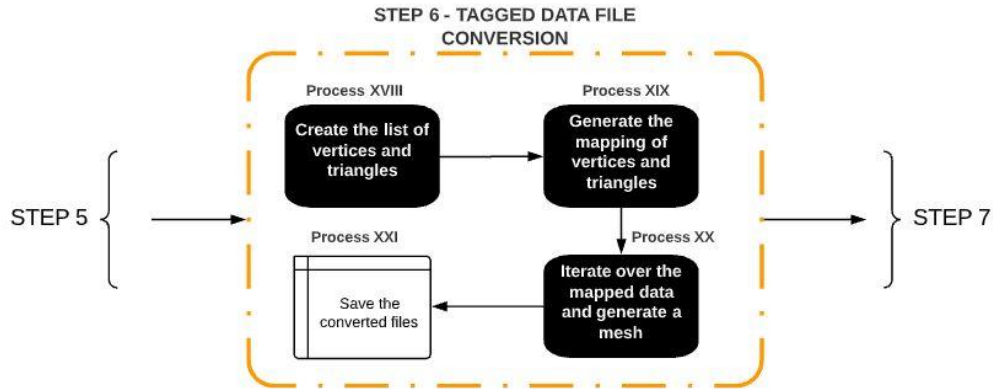


Figure 3.17. Step 6 Algorithm, DXF File Conversion



Figure 3.18. Examples of the Image of the OBJ Files Generated

3.3.3.2.8 Step 7- IFC File Generation

In this step, the encrypted bin file generated in Step 6 is utilized in generating the IFC output file. Figure 3.19 displays the algorithm for generating the IFC output file. *Process XXII* reads the encrypted bin file data for the projections. *Process XXIII* separates the data for the vertices and triangles. *Process XXIV* iterates over the separated data and appends the string values as specified in the ISO STEP standard. In developing the IFC output files, this manuscript employed python to read the bin data file and append the values to the corresponding representation in the STEP ISO-10303-21 standard format. *Process XXV* saves the generated files (Figure 3.20). Figure 3.20 displays a partial example of the output generated to illustrate the steps. These output files can be imported into various IFC viewers. Figure 3.21 displays an example of the IFC files for Bridges C, D, and E in BIMvision (BIMvision 2020).

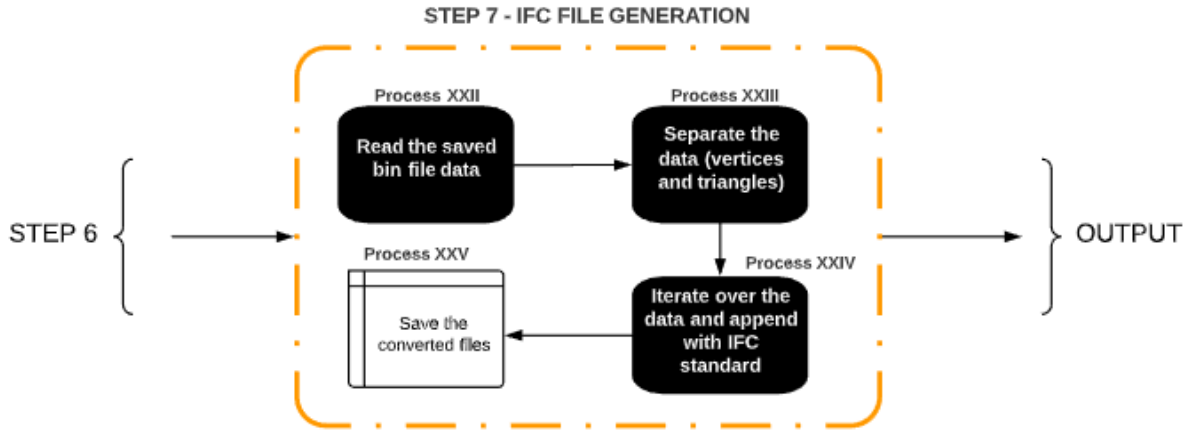


Figure 3.19. Step 7 Algorithm, IFC File Generation

```

1 ISO-10303-21;
2 HEADER;
3 FILE_DESCRIPTION('ViewDefinition [CoordinationView]', '2:1');
4 FILE_NAME('test2.blender.ifc', '2020-09-16T15:55:46+03:00', (), (), 'IfcOpenShell 0.6.0b0', 'BlenderBIM 0.0.200912', 'Nobody');
5 FILE_SCHEMA('IFC4');
6 ENDSPEC;
7 DATA;
8 #1=IFCSIUNIT(*, .LENGTHUNIT., $, .METRE.);
9 #2=IFCSIUNIT(*, .AREAUNIT., $, .SQUARE_METRE.);
10 #3=IFCSIUNIT(*, .VOLUMEUNIT., $, .CUBIC_METRE.);
11 #4=IFCUNITASSIGNMENT((#1, #2, #3));
12 #5=IFCARTESIANPOINT((0., 0., 0.));
13 #6=IFCDIRECTION((0., 0., 1.));
14 #7=IFCDIRECTION((1., 0., 0.));
15 #8=IFCAXIS2PLACEMENT3D(#5, #6, #7);
16 #9=IFCGEOMETRICREPRESENTATIONCONTEXT($, 'Model', 3, 1.E-05, #8, $);
17 #10=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body', 'Model', *, *, *, #9, $, .MODEL_VIEW., $);
18 #11=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Box', 'Model', *, *, *, #9, $, .MODEL_VIEW., $);
19 #12=IFCGEOMETRICREPRESENTATIONCONTEXT($, 'Plan', 2, 1.E-05, #8, $);
20 #13=IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Annotation', 'Plan', *, *, *, #12, $, .PLAN_VIEW., $);
21 #14=IFCPROJECT('2FUIkSYDEsmDcYraqr', $, 'My Project', $, $, $, ($9, #12), #4);
22 #25=IFCINDEXEDPOLYGONALFACE((5, 8, 6));
23 #30=IFCINDEXEDPOLYGONALFACE((1, 3, 5));
24 #31=IFCINDEXEDPOLYGONALFACE((3, 7, 5));
25 #32=IFCINDEXEDPOLYGONALFACE((1, 2, 3));
26 #33=IFCINDEXEDPOLYGONALFACE((2, 4, 3));
27 #34=IFCINDEXEDPOLYGONALFACE((2, 6, 8));
28 #35=IFCINDEXEDPOLYGONALFACE((2, 8, 4));
29 #36=IFCINDEXEDPOLYGONALFACE((1, 5, 2));
30 #37=IFCINDEXEDPOLYGONALFACE((2, 5, 6));
31 #38=IFCINDEXEDPOLYGONALFACE((13, 16, 14));
32 #39=IFCINDEXEDPOLYGONALFACE((13, 15, 16));
33 #40=IFCINDEXEDPOLYGONALFACE((9, 11, 13));
34 #41=IFCINDEXEDPOLYGONALFACE((11, 15, 13));
35 #42=IFCINDEXEDPOLYGONALFACE((9, 10, 11));
36 #43=IFCINDEXEDPOLYGONALFACE((10, 12, 11));
37 #44=IFCINDEXEDPOLYGONALFACE((10, 14, 16));
38 #45=IFCINDEXEDPOLYGONALFACE((10, 16, 12));
39 #46=IFCINDEXEDPOLYGONALFACE((11, 12, 16));
40 #47=IFCINDEXEDPOLYGONALFACE((11, 16, 15));
41 #48=IFCINDEXEDPOLYGONALFACE((9, 13, 10));
42 #49=IFCINDEXEDPOLYGONALFACE((10, 13, 14));
43 #50=IFCINDEXEDPOLYGONALFACE((21, 24, 22));
44 #51=IFCINDEXEDPOLYGONALFACE((21, 23, 24));
45 #52=IFCINDEXEDPOLYGONALFACE((17, 19, 21));
46 #53=IFCINDEXEDPOLYGONALFACE((19, 23, 21));
47 #54=IFCINDEXEDPOLYGONALFACE((17, 18, 19));
48 #55=IFCINDEXEDPOLYGONALFACE((18, 20, 19));
49 #56=IFCINDEXEDPOLYGONALFACE((18, 22, 24));
50 #57=IFCINDEXEDPOLYGONALFACE((18, 24, 20));
51 #58=IFCINDEXEDPOLYGONALFACE((19, 20, 24));
52 #59=IFCINDEXEDPOLYGONALFACE((19, 24, 23));
53 #60=IFCINDEXEDPOLYGONALFACE((17, 21, 19));
54 #61=IFCINDEXEDPOLYGONALFACE((18, 21, 22));
55 #62=IFCINDEXEDPOLYGONALFACE((29, 32, 30));
56 #63=IFCINDEXEDPOLYGONALFACE((25, 27, 29));
57 #64=IFCINDEXEDPOLYGONALFACE((27, 31, 29));
58 #65=IFCINDEXEDPOLYGONALFACE((25, 26, 27));
59 #66=IFCINDEXEDPOLYGONALFACE((26, 28, 27));
60 #67=IFCINDEXEDPOLYGONALFACE((26, 30, 32));
  
```

Figure 3.20. Partial Output (IFC File)

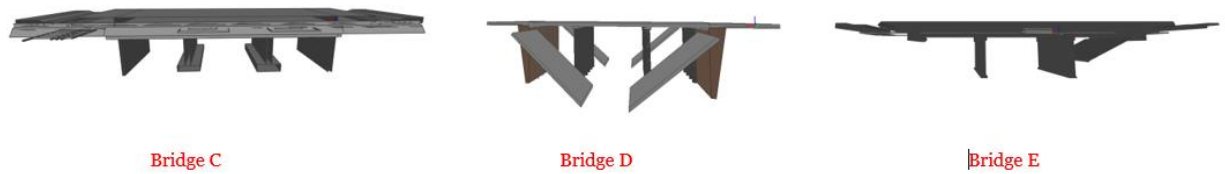


Figure 3.21. IFC Output Files Imported in BIMvision

3.3.3.2.9 Summary of Manual Efforts Involved in the Proposed Component

There are processes in three steps of the proposed framework that involve some manual operations. These steps are: (1) Step 1 – PDF Import: (2) Step 3 – Raster Graphics File Selection; and (3) Processes XIII and XIV – coordinate entries in Step 5. Similar to most BIM-based construction management platforms where users import the required model or file into the platform, the proposed component requires the user to import the PDF bridge plans into the system. This process is the first manual operation. Secondly, the process of selecting the raster graphics file (Step 3) requires a user to manually select the raster graphics sheet(s) from the raster graphics file generated in Step 2. The last set of manual operations required in the proposed component is the entry of coordinates in Step 5. A user is required to manually input the minimum and maximum entry coordinates (X, Y) of each projection into the system. Because these manual efforts are minor, this component is expected to greatly reduce the needed manual efforts in such documentation tasks when comparing to a pure manual method.

3.3.4 Experimental Results and Discussion

This section presents the results and analysis of the experiments performed for evaluating the proposed component. Three quantitative metrics were used: (1) mean distance between the point clouds of two compared models; (2) standard deviation from the Gaussian mean between point clouds of the 3D superimposed surfaces; and (3) the time it took to generate the models using the developed algorithms compared to a pure manual method. The results are summarized in Table 3.1. For metric 3, time spent using manual creation, both designers' average time was calculated and tabulated. Figure 3.22 displays the Gaussian distribution histogram for Bridge C generated from CloudCompare. CloudCompare utilizes the Hausdorff Distance theorem to compute the geometric difference between two 3D models. In comparing the two 3D models (developed Bridge

models using manual method and developed Bridge models using the developed algorithms), the surface change is estimated as a distance between two homologous points (for each point in Bridge models using the developed algorithms, the closest point is defined in Bridge models using Revit). Figure 3.23 displays the Cloud-to-Cloud comparison in CloudCompare. Tables 3.2 tabulates the partial results of the measurements of deviations in the distances recorded in Bridges C and D. For each bridge, there were multiple classes (cloud to cloud comparisons) generated; Bridge C produced 649 classes while Bridge D produced 546 classes as results. As shown in Table 3.1, the relatively low standard deviation indicates that the data points are close to the mean distances between both models; hence the developed algorithms generated models with a quality similar to that from the current state of the art pure manual practice. However, in comparing the time it took both methods to generate the 3D model, the developed algorithms took a user an average of 0.1785 hours (10 mins 43 secs) while the current state of the art manual practice took a user an average of 5 hrs. (300 mins). This shows that the developed algorithms using the proposed component utilized only 3.33% of the time it took using the current state of the art manual method to generate a 3D model.

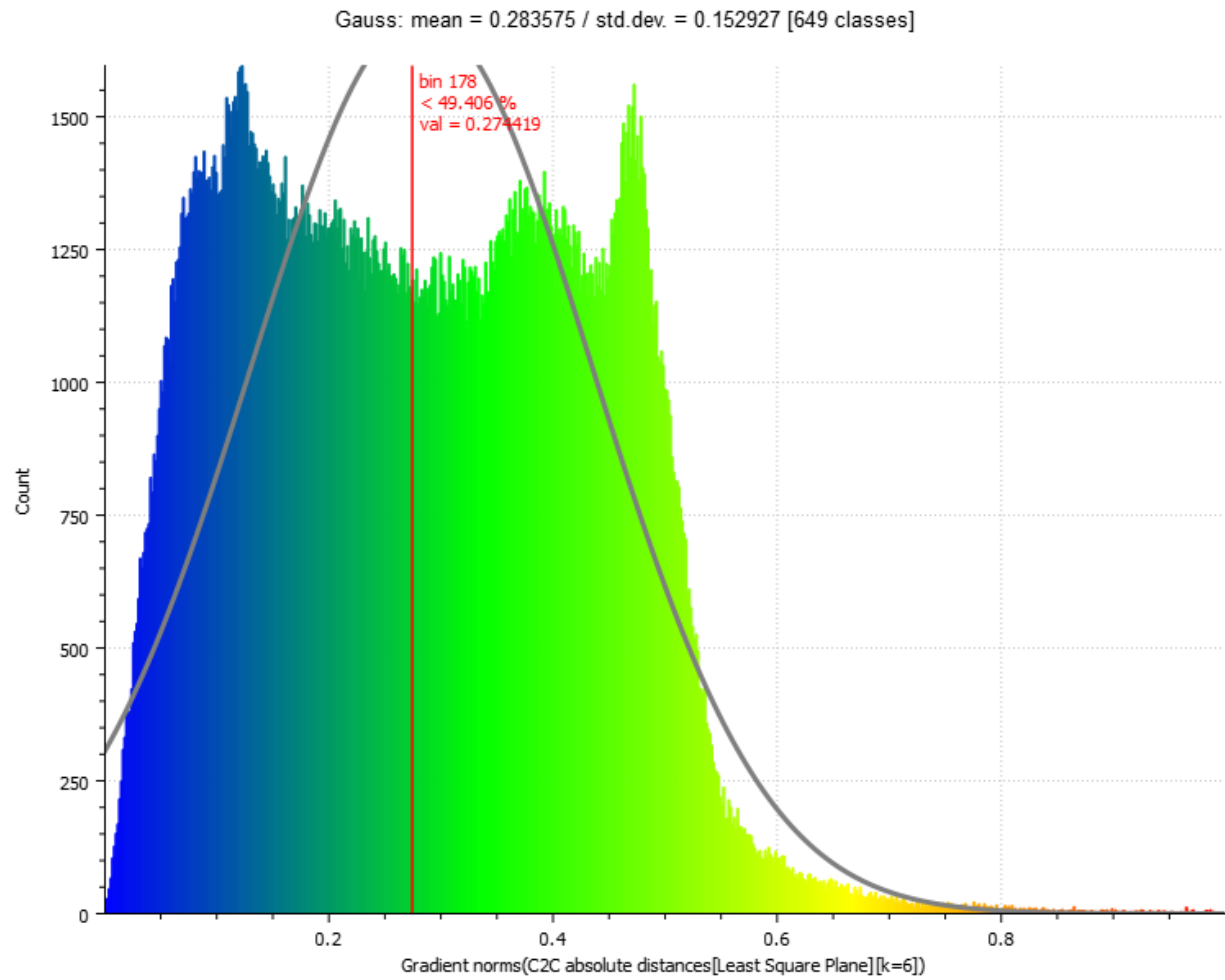


Figure 3.22. CloudCompare Statistical Test Distribution Histogram

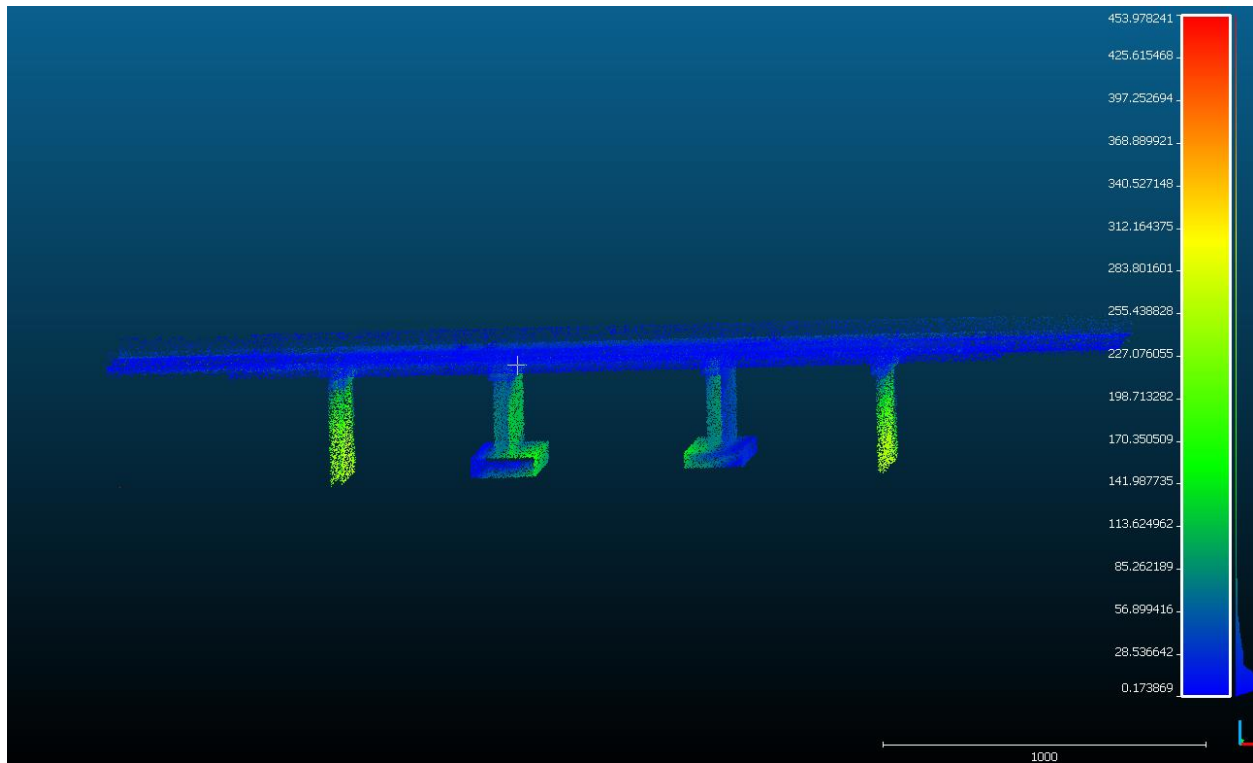


Figure 3.23. Cloud-to-Cloud distances (Example of Results – Bridge C)

Table 3.1. Experimental Results (Summary)

Bridge	Mean Distance	Standard Deviation	Time (Revit) hrs.	Time (Developed Algorithms) hrs.
C	0.283575	0.152927	5	0.1667
D	0.312375	0.216435	5.5	0.2167
E	0.246914	0.186235	4.8	0.1667
F	0.266784	0.221548	5.3	0.1667
G	0.291547	0.156485	4.3	0.1875
H	0.219265	0.101251	5.1	0.1667
Avg	0.270077	0.17248	5	0.1785

Table 3.2. Partial Experimental Results (Calculated Deviation in Distances)

Bridge C				Bridge D			
Class	Value	Class Start	Class End	Class	Value	Class Start	Class End
1	5626	0.000000000000	0.855101598613	1	8	0.000334584	0.000604521
2	5650	0.855101598613	1.710203197227	2	26	0.000604521	0.000874459
3	5600	1.710203197227	2.565304795840	3	69	0.000874459	0.001144397
4	5629	2.565304795840	3.420406394453	4	199	0.001144397	0.001414335
5	5611	3.420406394453	4.275507993066	5	314	0.001414335	0.001684272
6	5667	4.275507993066	5.130609591680	6	527	0.001684272	0.00195421
7	5643	5.130609591680	5.985711190293	7	804	0.00195421	0.002224148
8	5550	5.985711190293	6.840812788906	8	1067	0.002224148	0.002494086
9	5453	6.840812788906	7.695914387519	9	1409	0.002494086	0.002764023
10	5421	7.695914387519	8.551015986133	10	1774	0.002764023	0.003033961
11	5221	8.551015986133	9.406117584746	11	2312	0.003033961	0.003303899
12	5024	9.406117584746	10.261219183359	12	2667	0.003303899	0.003573837
13	4963	10.261219183359	11.116320781972	13	2887	0.003573837	0.003843774
14	4920	11.116320781972	11.971422380586	14	3425	0.003843774	0.004113712
15	4848	11.971422380586	12.826523979199	15	3841	0.004113712	0.00438365
16	4601	12.826523979199	13.681625577812	16	4169	0.00438365	0.004653588
17	4312	13.681625577812	14.536727176425	17	4570	0.004653588	0.004923525
18	3909	14.536727176425	15.391828775039	18	4917	0.004923525	0.005193463
19	3743	15.391828775039	16.246930373652	19	5186	0.005193463	0.005463401
20	3601	16.246930373652	17.102031972265	20	5534	0.005463401	0.005733339

3.4 Conclusions, Contributions, Limitations, and Recommendations for Future Research

3.4.1 Conclusions for the Proposed Modeling Component

In this dissertation, a new component was proposed to generate 3D bridge information models and bridge IFC output files from 2D bridge plans. The proposed component addresses the gap in the processing/conversion of traditional 2D bridge drawings into BIMs. The component can be utilized to develop algorithms that semi-automatically: (1) convert the 2D bridge plans in PDF file format to 3D information models; and (2) further converts the 3D information models to industry foundation classes (IFC) files. The proposed component was utilized in developing algorithms that were tested on six bridge projects in Indiana. The performance metrics results

indicated that the proposed component is effective and saves time and effort in generating 3D information models/IFC output files. Experimental results for the 3D model generation showed that the developed framework can be utilized in developing algorithms that can generate 3D models and IFC output files from Portable Document Format (PDF) bridge drawings in a semi-automated fashion. The developed algorithms utilized 3.33% of the time it took using the current state of the art method to generate a 3D model and the generated models were of comparative quality.

3.4.2 Contributions of the Proposed Modeling Component

This part contributes to the body of knowledge in two main ways: First, this study proposed a new component for developing algorithms that automate the generation of 3D bridge information models from 2D bridge plans to support BIM-based computations for bridges. In comparison to the current state-of-the-art practices in the industry that require the design/generation of these 3D information models manually, the developed algorithms significantly reduced the amount of manual effort and time involved in processing 2D bridge plans. Secondly, the developed algorithms using the proposed component further export the generated 3D information model to IFC output files. This improves the interoperability across platforms, stakeholders, and processes in the AEC industry.

3.4.3 Limitations of the Proposed Modeling Component

Four main limitations are acknowledged. Firstly, the developed algorithms were only tested on reinforced concrete slab beam bridges. Secondly, in the development of the algorithms, the configuration was mainly focused on the bridge objects. Other objects in a complete bridge plan, such as the topography and site work, were excluded. Thirdly, the proposed component involves some manual operations (although minor) in generating the 3D model. Fourthly, the current method assumes the required projections are usually contained in the plan sheet(s). In future work, the component would be extended to: (1) fully automate the generation of 3D information models; (2) process other bridge forms such as arch bridges, truss bridges, etc.; (3) process and generate

other information needed such as the topography and site work and (4) automate the process of searching for the required projections from the bridge PDF files.

CHAPTER 4. QUANTIFICATION COMPONENT – A DATA-DRIVEN REVERSE ENGINEERING ALGORITHM DEVELOPMENT (D-READ) METHOD FOR DEVELOPING INTEROPERABLE QUANTITY TAKEOFF ALGORITHM USING IFC-BASED BIM

A version of this chapter has been published in the Journal of Computing for Civil Engineering.

Akanbi, T., Zhang, J., and Lee, Y-C. (2020). "Data-Driven Reverse Engineering Algorithm Development Method for Developing Interoperable Quantity Takeoff Algorithms Using IFC-Based BIM." *Journal of Computing in Civil Engineering*, 34(5): 04020036. DOI: 10.1061/(ASCE)CP.1943-5487.0000909. "With permission from ASCE"

This chapter presents a new Data-driven Reverse Engineering Algorithm Development (D-READ) method, an iterative method to generate QTO algorithms that can cover various (and eventually all) types of BIM representations in IFC. The proposed method enables the development of QTO algorithms for IFC-based BIMs resulting from different BIM authoring tools/workflows and, therefore, enhances BIM-based QTO robustness. 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.

4.1 IFC-Based QTO Methods

There are different QTO solutions geared towards solving specific issues in the construction industry. In the 2D realm, some tools enable construction professionals to digitally perform QTO using 2D drawings such as eTakeoff, On-Screen Takeoff, PlansSwift, etc. In using 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 to their wide adoption due to the interoperability problem. Such QTO solutions require importing a building design (mostly 3D) 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 different types of research investigating information extraction from IFC-based BIM for various purposes (Zhang and El-Gohary 2016, 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, current methods do not address QTO from building information models (BIMs) created from different BIM authoring tools/workflows. Such a capability is critical to enable wider adoption of BIM-based QTOs.

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 differently.

4.2 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 dissertation proposed 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. In this dissertation, it was named 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 (Figure 4.1). 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.

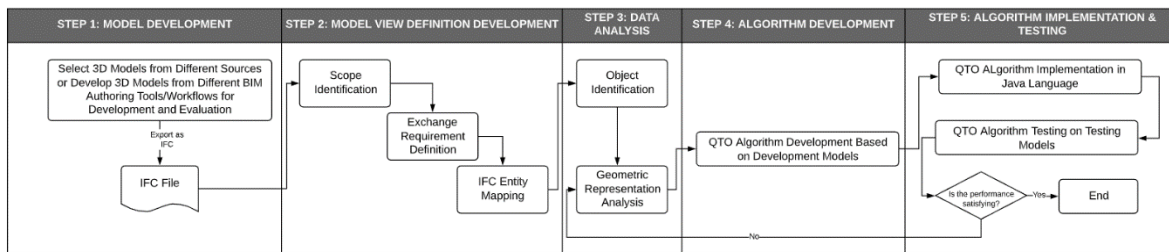


Figure 4.1. Proposed Data-Driven Reverse Engineering Algorithm Development (D-READ) Method

An illustration of how developed QTO algorithms resulting from the D-READ method should be applied is shown in Figure 4.2. 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 extract the quantities of building components from it. The developed algorithms achieve these by analyzing the model-specific geometric representations of building objects based on arbitrary choices made in the proprietary BIM authoring tools/workflows under the constraints set by IFC schemas.

The proposed D-READ method takes IFC models as input, 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. For 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 Revit. The details of the five steps of the D-READ method are described in the following subsections, respectively.

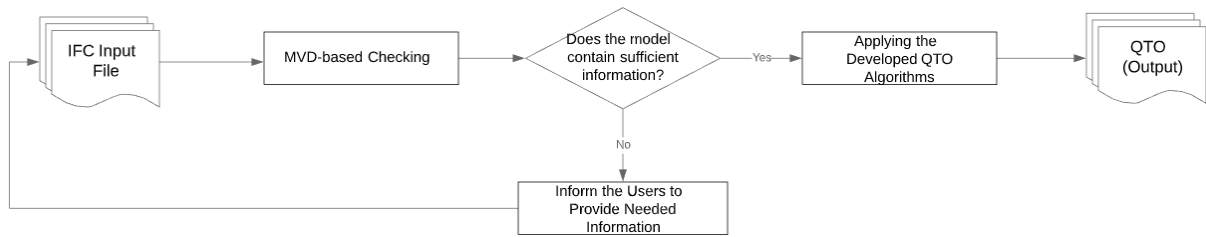


Figure 4.2. An Illustration of the Application of Developed QTO Algorithms

4.2.1 Step 1- Model Development

In this step, the model data used in QTO algorithm development and evaluation are selected or generated, including development data, testing data for algorithm development, and evaluation data for algorithm evaluation. The development 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, 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 development model, a testing model, and an evaluation model.

4.2.2 Step 2- Model View Definition (MVD) Development

In the development of the MVD, there are the following three sub-steps: scope identification, exchange requirement definition, and IFC entity mapping. In an AEC project, different user groups (e.g., clients, architects) require 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

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 are 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.

4.2.3 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:

4.2.3.1 Object Identification

The object identification step is necessary for deriving the needed information required in developing the QTO algorithms. As shown in Figure 4.3, 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 determines 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*,” “*ObjectPlacemenet*,” “*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 should be in the list and what each type of representation will be. Thus, not until development data is analyzed can it 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 Figure 4.4. 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.

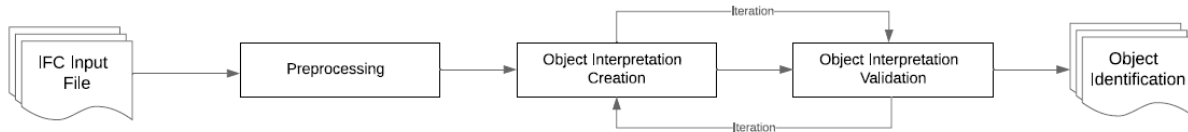


Figure 4.3. Object Identification Processes

4.2.3.2 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. The analysis result is used to create data tracing patterns for QTO purposes. To illustrate this process, the tracing patterns in retrieving the attributes of a wall, opening, floor and materials would be explained below. This target-specific IFC data representation analysis can accommodate different definitions and measurements of quantities based on the cost estimation need. For example, the height of a wall may be defined ceiling to ceiling or floor to ceiling.

4.2.3.2.1 Wall Component - extracting the length, width, and height attributes

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 (Figure 4.4). 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. This target-specific IFC data representation analysis can accommodate different definitions and measurements of quantities based on the cost estimation need. For example, the height of a wall may be defined ceiling to ceiling or floor to ceiling.

As shown in Figure 4.4, *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 *BodyRepresentationOne* 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. A decision is made at this point to check if the *SweptArea* uses “IFCRECTANGLEPROFILE,” i.e., to check if it is a rectangular-shaped wall.

Rectangular-shaped wall- 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. At this point, the quantity measures for a rectangular-shaped wall using “*SweptSolid*” *BodyRepresentation* would be successfully extracted.

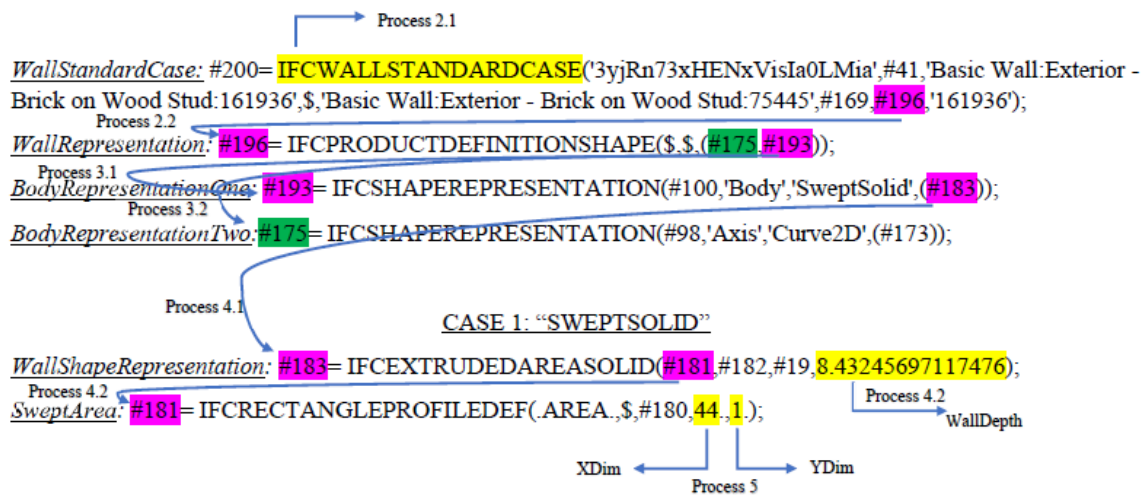


Figure 4.4. Tracing Pattern of a “SweptSolid” Representation of a Rectangular Wall

Curved Wall- If the *SweptArea* uses “IFCARBITRARYCLOSEDPROFILEDEF,” as shown in Figure 4.5, then *Process 6.1* extracts the third attribute of *SweptArea*, which is an “IFCCOMPOSITECURVE” and writes it into *OuterCurve*. *Process 6.1* further extracts the first element of the first attribute of *OuterCurve*, which is an “IFCCOMPOSITECURVESEGMENT” and writes it into *WallSegmentOne*. The third attribute from *WallSegmentOne* is extracted in *Process 6.2* to give an “IFCTRIMMEDCURVE” and writes it into *ParentCurve*. *Process 7.1* extracts the value of the first element of the second attribute into *TrimOne* and *Process 7.2* extracts the value of the second element of the second attribute into *TrimTwo*. *Process 8* extracts all lines containing “IFCCIRCLE” from an array list created. The array list is created by reading all entities and their attributes into a java array list to perform the different extraction processes for the required variables. *Process 8* extracts the lines containing “IFCCIRCLE” from the array list and writes these entities into *Circle*. *Process 8* further extracts the second attributes of *Circles* into *Radius1*, *Radius2* and *Radius3*, respectively. *Process 9* computes the length of the wall using Equation [I]. *Process 10* computes the width of the wall using Equation [II]. In Equation [I], (*L*) is the length of the wall; *Radius1* denotes the radius of the center curve; *TrimOne* is the first trimming point of the curve; and *TrimTwo* is the second trimming point of the curve. In Equation [II], (*W*) is the width of the wall; *Radius2* denotes the radius of the inner curve; and *Radius3* denotes the radius of the outer curve. At this point, the quantity measures for a curved wall “*SweptSolid*” *BodyRepresentation* would be successfully extracted.

$$(L) = 2 * Pi * Radius1 * \left\{ 1 - \left[\frac{TrimOne - TrimTwo}{360} \right] \right\} \quad [I]$$

$$(W) = Radius2 - Radius3 \quad [II]$$

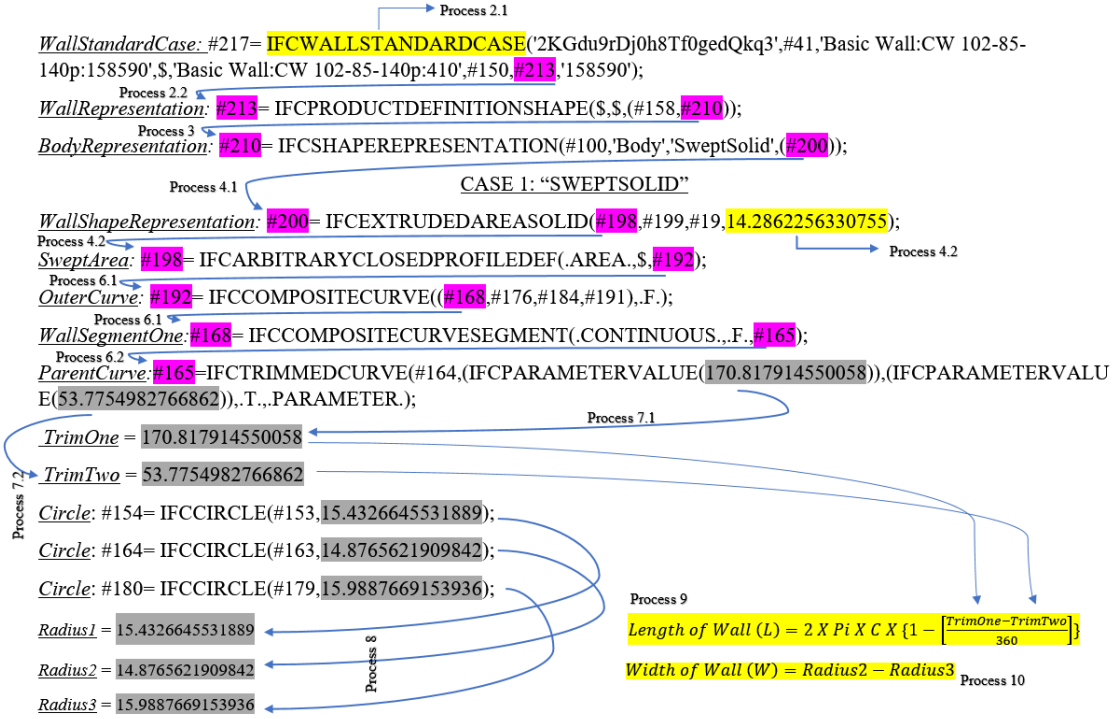


Figure 4.5. Tracing Pattern of a "SweptSolid" Representation of a Curved Wall

CASE 2- "Clipping" BodyRepresentation

As shown in Figure 4.6, if *BodyRepresentation* contains "Clipping," i.e., it is using "Clipping" type of shape representation, then *Process 4.3* extracts the first element of the fourth attribute of *BodyRepresentation*, which is an "IFCBOOLEANCLIPPINGRESULT," and writes it into *WallShapeRepresentation*. *Process 4.4* extracts the second attribute of *WallShapeRepresentation*, which is an "IFCEXTRUDEDAREASOLID," into *FirstOperand*. *Process 4.4* further extracts the fourth attribute of *FirstOperand*, which is the *WallDepth* (i.e., the height of the wall). *Process 4.5* extracts the first attribute of *FirstOperand*, which is an "IFCRECTANGLEPROFILEDEF," into *SweptArea*, *Process 5* extracts the fourth and fifth attributes of *SweptArea* as *XDim* (i.e., the length of the wall) and *YDim* (i.e., the width of the wall), respectively. At this point, the quantity measures for a rectangular-shaped wall "Clipping" *BodyRepresentation* would be successfully extracted.

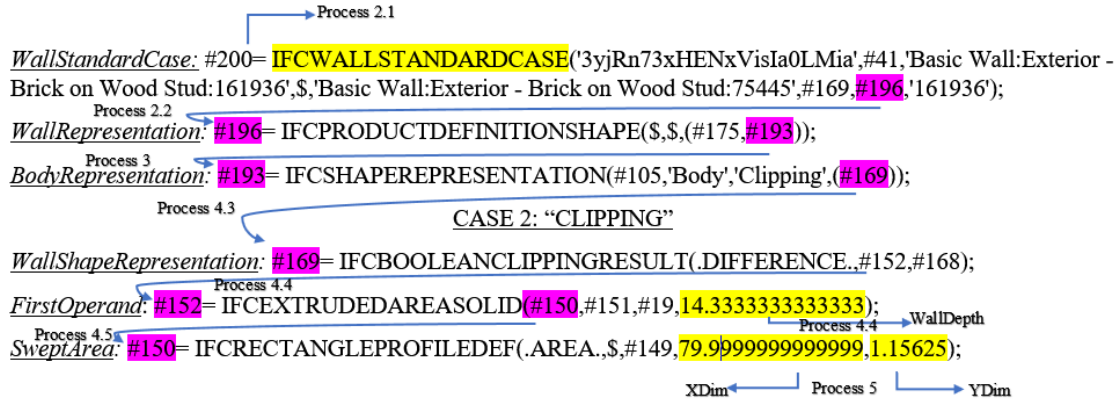


Figure 4.6. Tracing Pattern of a “Clipping” Representation of a Rectangular Wall

Curved Wall- Similar to the method described under curved wall “SweptSolid” shape representation (Figure 4.5).

4.2.3.2.2 Opening Element - extracting the length, width, and height attributes

As shown in Figure 4.7, *Process 11.1* extracts all lines containing “IFCOPENINGELEMENT” into *OpeningElement*. *Process 11.2* extracts the seventh attribute of *OpeningElement*, an “IFCPRODUCTDEFINITIONSHAPE,” and writes it into *OpeningRepresentation*. *Process 12* extracts the first element of the third attribute of *OpeningRepresentation*, which is an “IFCSHAPE REPRESENTATION,” into *OpeningDefShape*. *Process 13.1* extracts the first element of the fourth attribute of *OpeningDefShape*, which is an “IFCEXTRUDEDAREASOLID,” into *OpeningItem*. *Process 13.2* extracts the fourth attribute of *OpeningItem* into *OpeningWidth*; *Process 13.2* further extracts the first attribute of *OpeningItem*, which is an “IFCRECTANGLEPROFILEDEF,” into “*OpeningArea*.” *Process 14* extracts the fourth and fifth attributes of *OpeningArea* into *XODim* and *YODim*, respectively. At this point, the quantity measures for an opening element would be successfully extracted.

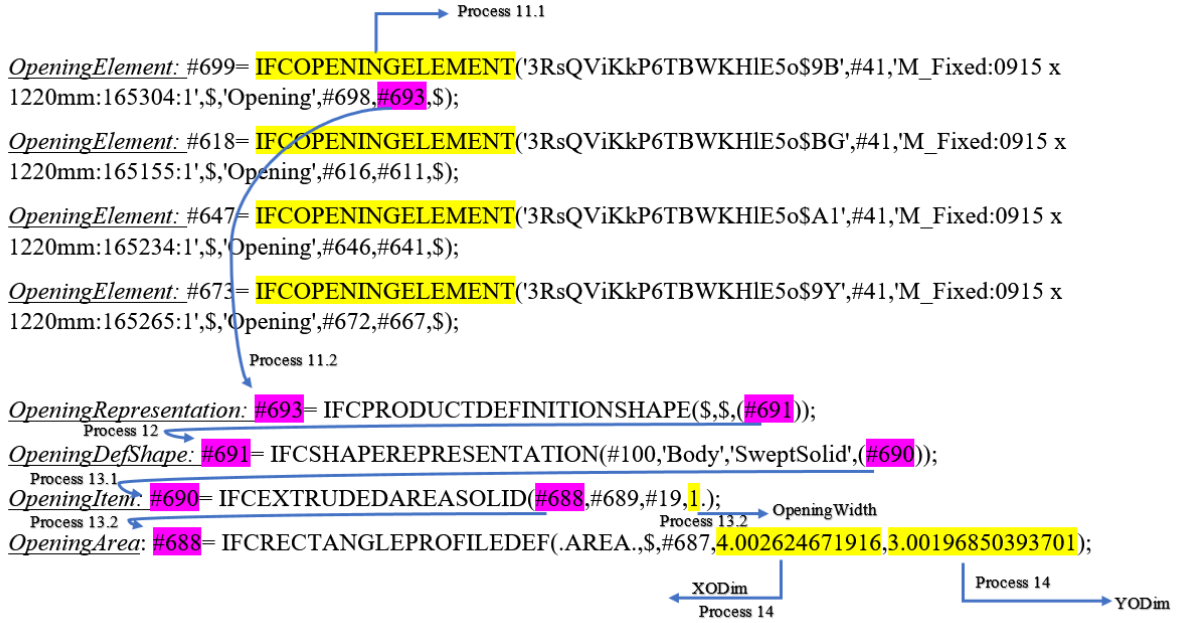


Figure 4.7. Tracing Pattern of an Opening

4.2.3.2.3 Floor Component - extracting the length, width, and thickness attributes

Figure 4.8 shows the processes involved in extracting the attributes of a rectangular-shaped “SweptSolid” geometric floor representation. As shown in Figure 4.8, *Process 2* performs a search for “IFCSLAB” entity and writes the found entity into *FloorSlab*. *Process 2* further extracts the seventh attribute of *FloorSlab*, an “IFCPRODUCTDEFINITIONSHAPE,” and writes it into *FloorRepresentation*. *Process 3* extracts the first element of the third attribute of *FloorRepresentation*, an “IFCSHAPEREPRESENTATION,” and writes it into *BodyRepresentation*. Similar to the wall component, at this level, depending on the type of *BodyRepresentation*, the corresponding case branches are activated. If *BodyRepresentation* contains “SweptSolid,” *Process 4.1* extracts the first element of the fourth attribute of *BodyRepresentation*, which is an “IFCEXTRUDEDAREASOLID,” into *FloorShapeRepresentation*. *Process 4.2* extracts the fourth attribute of *FloorShapeRepresentation* into the *FloorThickness* (i.e., the thickness of the floor). *Process 4.3* extracts the first attribute of *FloorShapeRepresentation*, which is an “IFCRECTANGLEPROFILEDEFAREA,” into *SweptArea*. *Process 5.1* extracts the fourth attribute of *SweptArea* as the *FloorLength* (i.e., the

length of the floor) and extracts the fifth attribute of the *SweptArea* as the *FloorWidth* (i.e., the width of the floor).

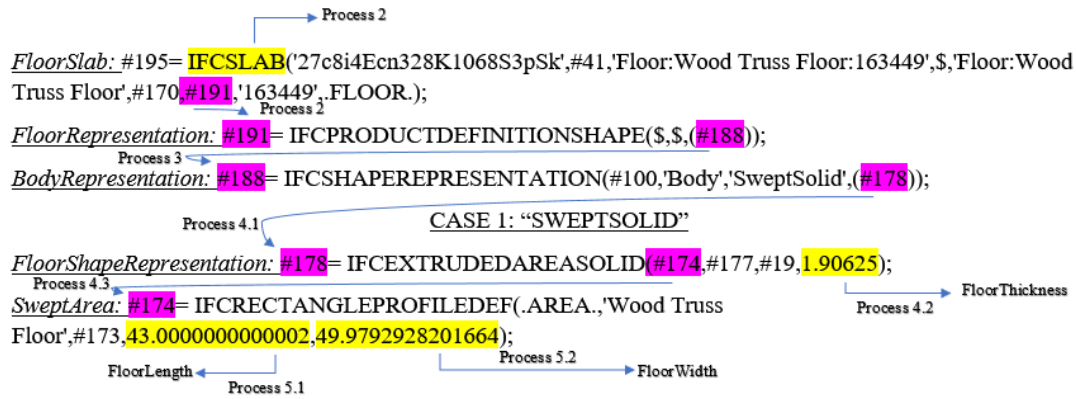


Figure 4.8. Tracing Pattern of a “SweptSolid” Representation of a Rectangular Floor

4.2.3.2.4 Material Layer Set - extracting the material list and thickness

As shown in Figure 4.9, *Process 15* extracts the line containing “IFCMATERIALLAYERSET” into *MaterialLayerSet*. *Process 16.1* extracts all elements of the first attribute of *MaterialLayerSet*, each of which is an “IFCMATERIALLAYER,” into “*MaterialLayerThickness*.” *Process 16.2* extracts the second attribute of “*MaterialLayerThickness*” into “*MThickness*,” which is the thickness of the material layer. The first attribute of “*MaterialLayerThickness*,” which is “IFCMATERIAL,” is further extracted into “*MaterialLayerList*.” Each material in the “*MaterialLayerList*” is represented by a String label indicating the material type. At this point, all materials and their corresponding thicknesses are retrieved.

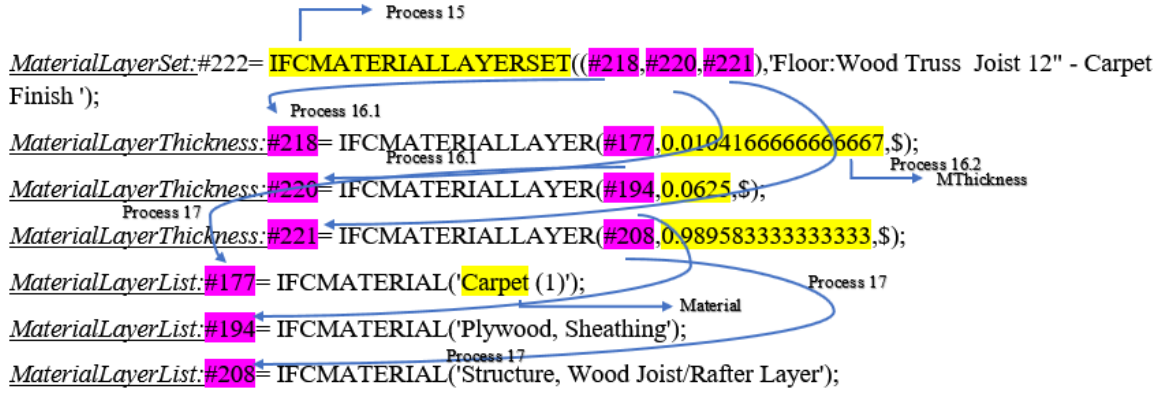


Figure 4.9. Tracing Pattern of the Material Layer Attributes

Similarly, the tracing patterns of the stairs 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) (Figure 4.10) and the second tracing pattern was used by Model E (Pattern S_2) (Figure 4.11). 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 (**ThreadLength**); (3) the number of risers (**RiserNumber**); and (4) the number of threads (**ThreadNumber**). These parameters are used in Equation [III] 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 [III] and [IV] to calculate the length of the flight (**FlightLength**). The main difference between these two tracing patterns of stairs is in the parameters used. While the *RelDefinesByProperties* in Pattern S_1 contained **ThreadDepth**, the *RelDefinesByProperties* in Pattern S_2 does not contain **ThreadDepth**.

$$\text{FlightLength} = \text{ThreadNumber} * \text{ThreadDepth} \quad \text{[III]}$$

$$\text{ThreadDepth} = 17.5'' - \text{RiserHeight} \quad \text{[IV]}$$

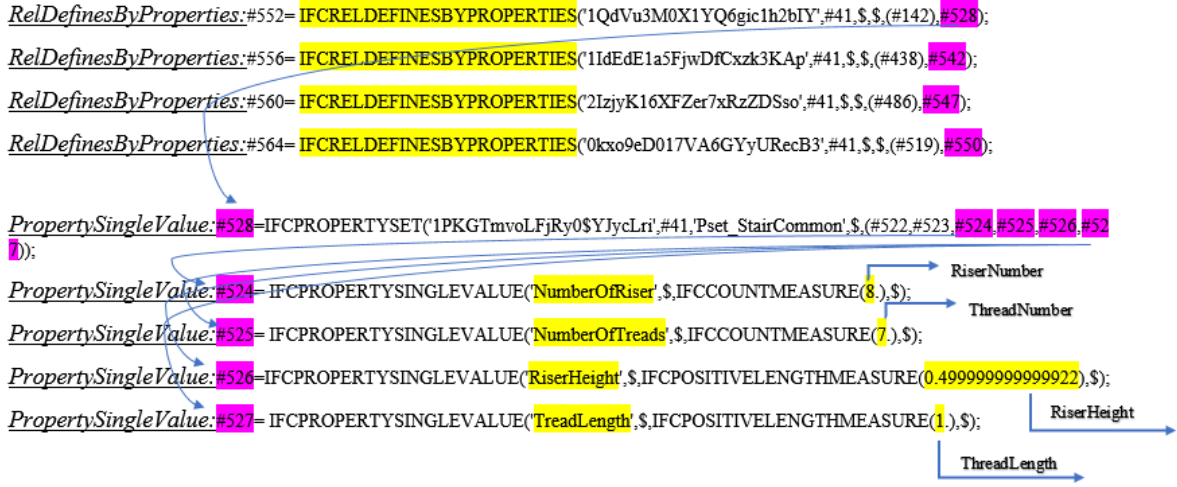


Figure 4.10. Tracing Pattern S_1 of Stairs

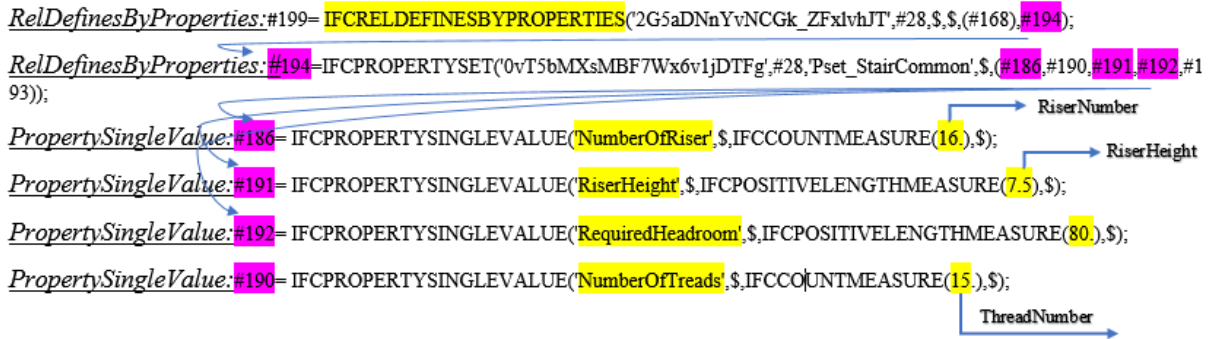


Figure 4.11. Tracing Pattern S_2 of Stairs

4.2.4 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.

4.2.5 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 (Figure 4.12). 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.

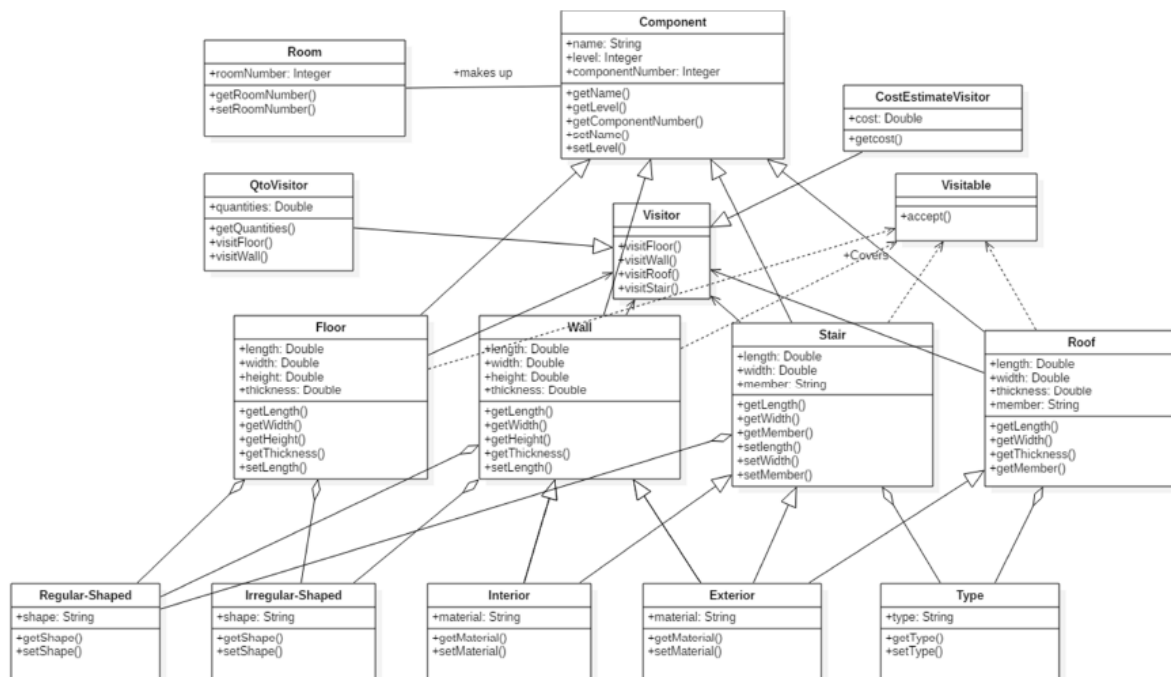


Figure 4.12. UML Class Diagram for the Algorithm Development

4.3 Experimental Testing and Evaluation

To test the effectiveness of the proposed D-READ method, the proposed method was tested on six building models. The details of the experiment are described in the following subsections.

4.3.1 Experimental Data

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

4.3.1.1 QTO Algorithm Development Using the D-READ Method

4.3.1.1.1 Step 1- Model Development

Training and testing data: Three models were created based on an apartment complex building project in Kalamazoo, Michigan. Hard copy architectural drawings were obtained from the project owner. In this dissertation, 3D models of the building were created by the author in three different BIM authoring tools according to the drawings to a BIM level of detail 300, namely, Revit 2019 (Model *A*), SketchUp 2019 (Model *C*), and ArchiCAD 2019 (Model *E*) (Figure 4.13). The building has 38 units made up of 44,192 sq.ft. at three floor levels; 32 two-bedroom units and 6 four-bedroom units. The 3D model data were further converted into IFC format through the built-in export functions in the BIM authoring tools. Models *A*, *C*, and *E* were used for development. The testing Models *B*, *D*, and *F*, are also shown in Figure 4.13. 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: Figure 4.14 shows the three BIM instance models developed for evaluation purposes. Similar to the development and testing data, the three models were created in the three BIM authoring tools. Models *G* and *I* were created in this dissertation using Autodesk Revit and GRAPHISOFT ArchiCAD, whereas Model *H* was retrieved from the online 3D Warehouse based on a residential apartment building created by Razin Kahn in Trimble SketchUp.

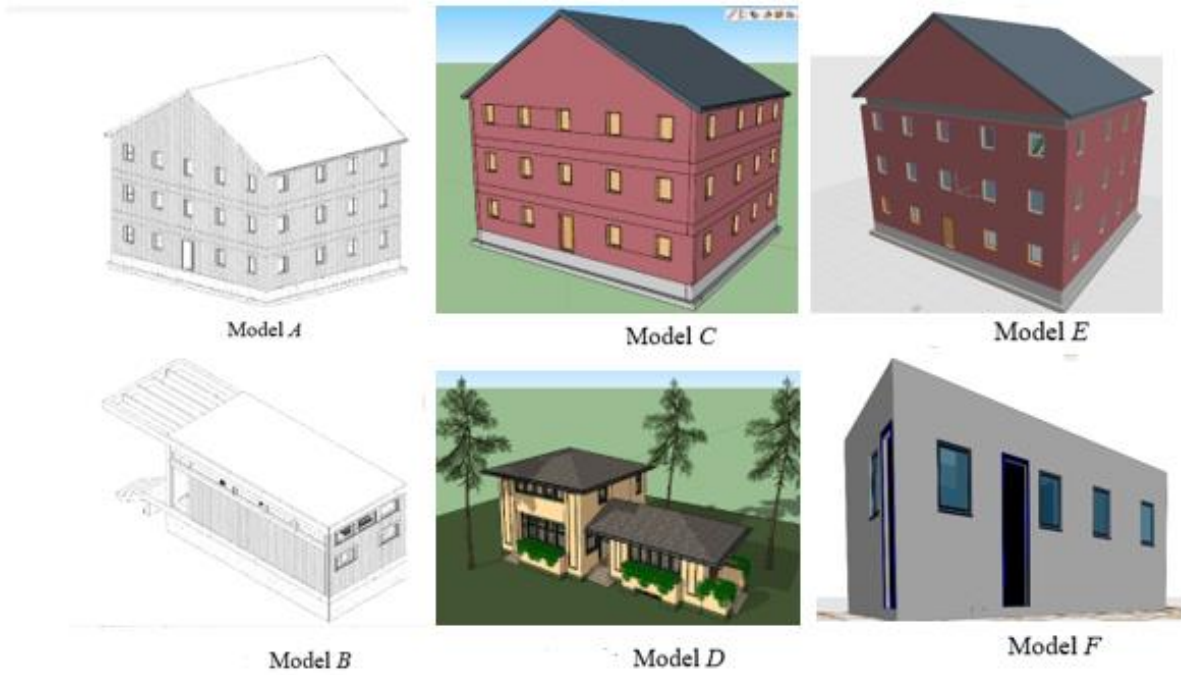


Figure 4.13. Visualization of the Training and Testing Data

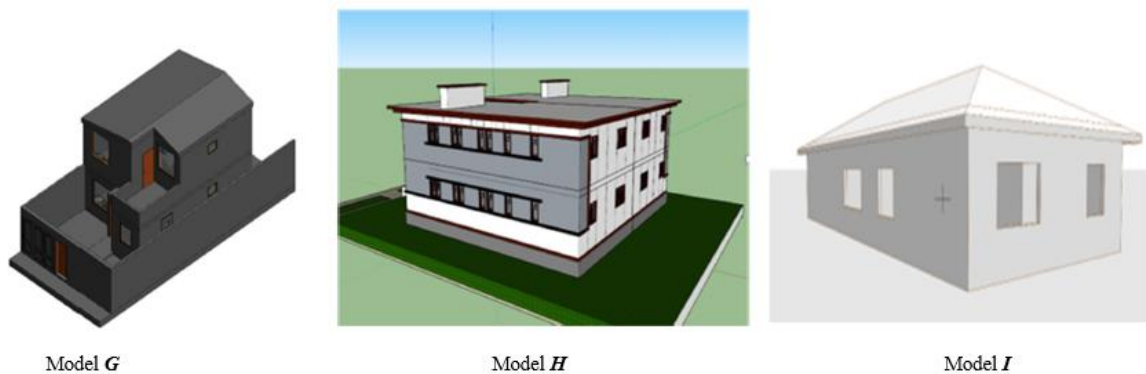


Figure 4.14. Visualization of the Evaluation Data

4.3.1.1.2 Step 2- Model View Definition Development

To develop the MVD, this manuscript utilized the ifcDOC tool (buildingSMART 2015), which is an open-source MVD creation tool for creating model views, concept templates, concept roots, and concept leaves (Figure 4.15).

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 Coordination View (CV) 2.0, 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 2.0, this dissertation defined rules for supporting the QTO exchange requirements needed for exporting the IFC 2X3 file corresponding to a subset of the CV 2.0 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 Figure 4.15, 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.

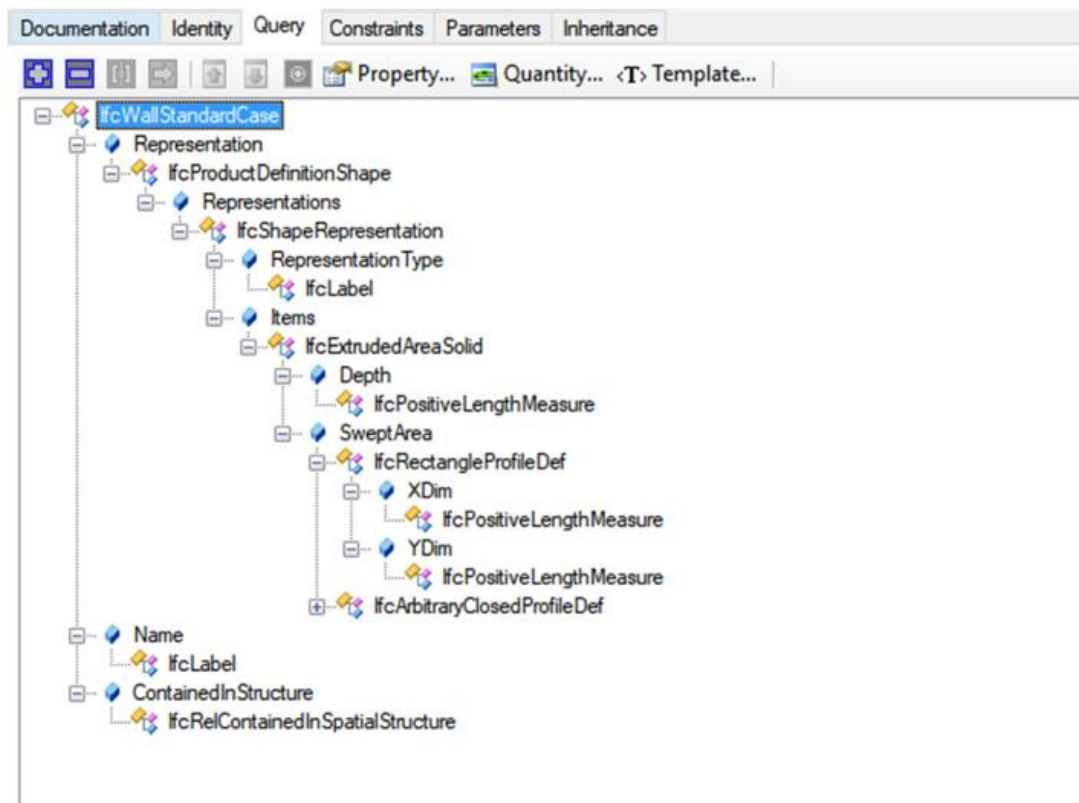


Figure 4.15. Exchange Requirement for an MVD Concept Wall in the ifcDOC Interface

Figure 4.16 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 Figure 4.2, the system would inform the users to provide the needed information); or (2) sufficient information (the IFC file is passed on to the developed QTO algorithms for data extraction). The validation results were extracted using jsoup Java HTML parser API (jsoup 1.11.3).

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]

Figure 4.16. An Output Report from an Example MVD Validation

4.3.1.1.3 Step 3- Data Analysis

Object Identification and Geometric Representation Analysis

Each object in the IFC files of the development 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 using IFC entities and attributes.

4.3.1.1.4 Step 4- Algorithm Development

The algorithms developed used tracing patterns in the IFC data structure and the component's geometric representations in the IFC development 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. Figure 4.17 shows the QTO algorithm developed for extracting the attributes of the wall, while Figure 4.18 shows the QTO algorithm for extracting the attributes of the floor from its geometric representation in an IFC file.

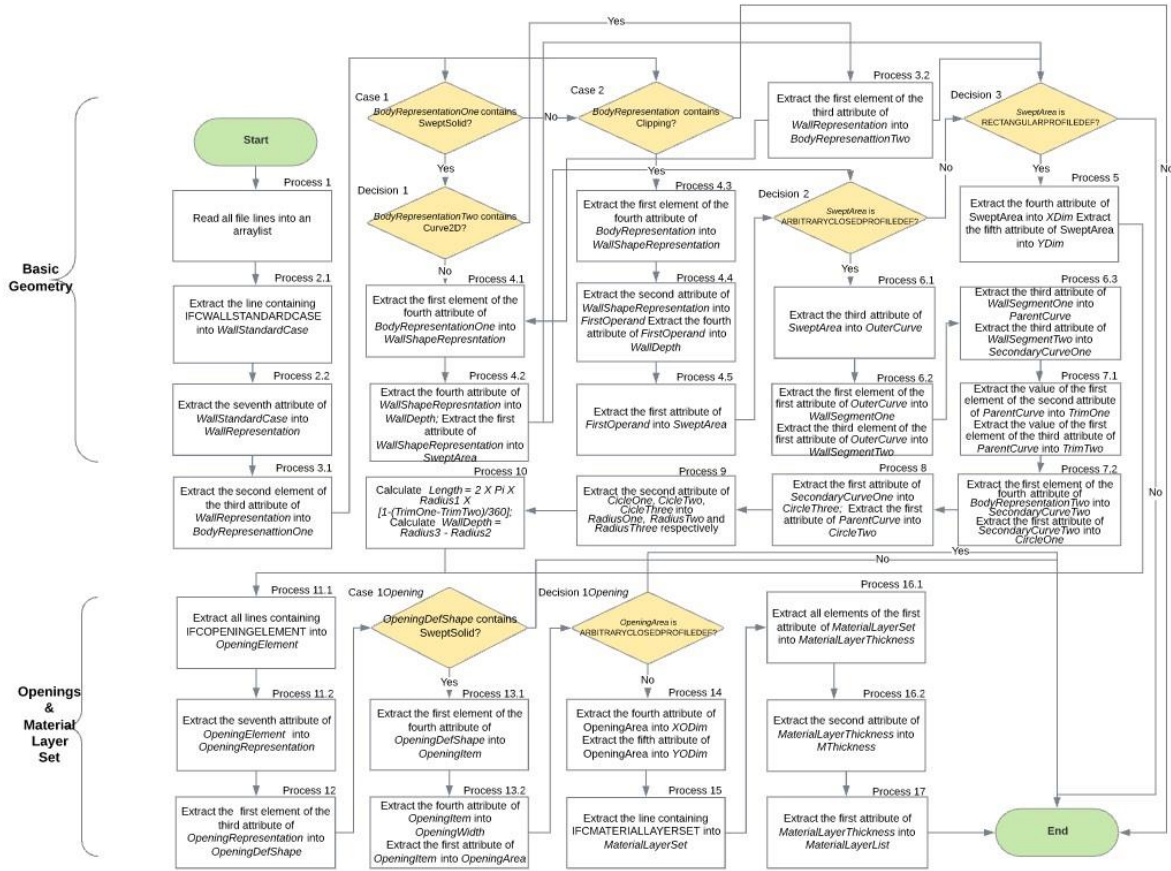


Figure 4.17. Flowchart of the QTO algorithms for a Wall

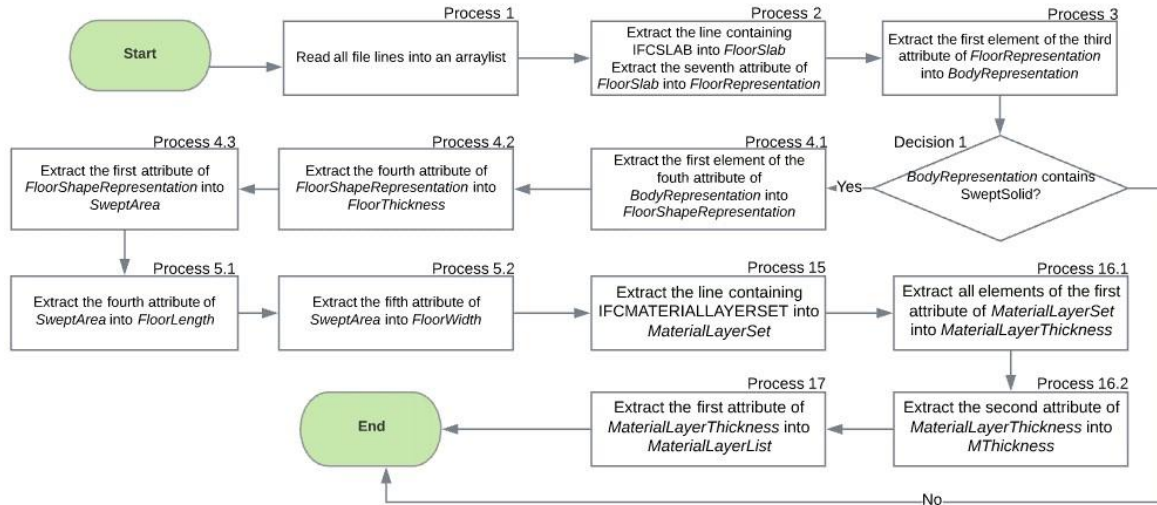


Figure 4.18. Flowchart of the QTO algorithms for a Floor

4.3.1.1.5 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. An example object of each type was selected from each of the building models, and the QTO results were tabulated in Table 4.1. Figure 4.19 shows an interface of output results of a wall instance (Model *B*) using the implemented QTO algorithms in Java.

Table 4.1. 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 ($^{\circ}$)
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	-	-	-	-	

```

workspace - Java - temitop/resources/database.properties - Eclipse
File Edit Navigate Search Project Express Run Window Help

New_configuration (8) [Java Application] 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:          0.4166666666666671
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
  
```

Figure 4.19. An Example Output Results from a Wall Instance Using the Implemented QTO Algorithms in Java

4.3.2 Experimental Evaluation Results and Discussion

4.3.2.1 Accuracy of results

To evaluate the accuracy of the QTO algorithms produced by the 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 this dissertation had access to only support models G and H. In contrast, the QTO algorithms developed using the D-READ method were successfully used to extract 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 4.2 and Table 4.3. Comparing 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 4.2. 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 Deviation (%)	13.0577	0.4167	7.4602 0%	97.4134	40.5922
Wall 2	Algorithm	20.6693	0.4167	7.9193	163.6868	68.2083
	Commercial Software Deviation (%)	20.6693	0.4167	7.9193 0 %	163.6868	68.2083
Wall 3	Algorithm	20.21	0.4167	7.1768	145.0437	60.4397
	Commercial Software Deviation (%)	20.21	0.4167	7.1768 0 %	145.0437	60.4397
Wall 4	Algorithm	6.7453	0.4167	7.4602	50.3213	20.9689
	Commercial Software Deviation (%)	6.7453	0.4167	7.4602 0 %	50.3213	20.9689
Floor	Algorithm	39.0420	17.2498	0.6561	673.4656	441.8924
	Commercial Software Deviation (%)	39.0420	17.2498	0.6561 0 %	673.4656	441.8924
Roof		Length (ft.)	Width (ft.)	Area (sq. ft.)	Volume (cu. ft.)	Slope (°)
	Algorithm	47.0	0.7916	2929.9364	2319.3349	30
Stair	Commercial Software Deviation (%)		0.7916	2929.9364 0 %	2319.3349	
		Riser height (ft.)	Thread Length (ft.)	Flight Length (ft.)	Stairs width (ft.)	
	Algorithm	0.4261	0.8202	14.7638	2.9528	
	Commercial Software Deviation (%)			0 %	2.9528	

Table 4.3. 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 Deviation (%)	21.1942	0.4167	9.0914 0%	159.6839	66.5350
Wall 2	Algorithm	20.3280	0.4167	8.8583	131.8486	54.9369
	Commercial Software Deviation (%)	20.3280	0.4167	8.8583 0 %	131.8486	54.9369
Wall 3	Algorithm	20.7283	0.4167	8.8583	177.6169	74.0070
	Commercial Software Deviation (%)	20.7283	0.4167	8.8583 0 %	177.6169	74.0070
Wall 4	Algorithm	14.5647	0.4167	8.8583	123.0178	51.2574
	Commercial Software Deviation (%)	14.5647	0.4167	8.8583 0 %	123.0178	51.2574
Floor	Algorithm	29.3044	21	1.0625	615.3937	653.8558
	Commercial Software Deviation (%)	29.3044	21	1.0625 0 %	615.3937	653.8558

4.3.2.2 Robustness of method

To evaluate the robustness of the proposed D-READ method, a comparison of the D-READ method in generating QTO components from various BIM authoring tools against different state-of-the-art commercial software was tabulated in Table 4.4. 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, respectively. As at the time of this research, this dissertation 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 is 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 4.4. Robustness evaluation

BIM platform	QTO tool compatibility			
	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	?	?	?

4.4 Conclusions, Contributions, Limitations, and Recommendations for Future Research

4.4.1 Conclusions for the Proposed Quantification Component

To establish interoperable QTO methods using BIMs created from different BIM authoring tools/workflows, a new D-READ method was developed in the dissertation which 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 development, three for testing, and three for evaluation. QTO algorithms were produced for walls, 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 the state-of-the-art QTO tools. The algorithms successfully extracted the quantities of the evaluation models consistent with the state-of-the-art tools. While none of the studied 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. Therefore, the D-READ method proposed in this study 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.

4.4.2 Contributions of the Proposed Quantification Component

This part 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 tools/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 accumulated into a comprehensive set to cover different objects in different construction projects. Secondly, the presented work extended the current available architectural MVD specifications to one that checks IFC instance files for architectural QTO purposes and leverages them in the new D-READ method. This is a pioneer research effort in systematically solving interoperability and automation of BIM-based QTO.

4.4.3 Limitations of the Proposed Quantification Component

Two main limitations are acknowledged: currently, the D-READ method produces QTO algorithms that could only address geometric representations observed in the development data. The QTO algorithms were currently tested on explicitly modeled elements (e.g., walls, slabs), unmodeled elements (e.g., scaffolding) were not tested. 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 the development data, and QTO algorithms for unmodeled elements will be tested.

CHAPTER 5. EXTRACTION COMPONENT - AUTOMATED DESIGN INFORMATION EXTRACTION FROM SPECIFICATIONS TO SUPPORT CONSTRUCTION COST ESTIMATION AUTOMATION

A version of this chapter has been submitted to the Automation in Construction Journal:

Akanbi, T., and Zhang, J. (2021). “Design Information Extraction from Construction Specifications to Support Cost Estimation.” *Automation in Construction*, submitted.

The background (5.1) appears in the following publication:

Akanbi, T., Zhang, J. (2020). “Automated Design Information Extraction from Construction Specifications to Support Wood Construction Cost Estimation.” *Proc., 2020 ASCE Construction Research Congress*, ASCE, Reston, VA, 658-666. DOI: 10.1061/9780784482889.069. "With permission from ASCE"

This chapter presents a new method that uses semantic modeling and natural language processing techniques in developing algorithms that automate the manual processes involved in: (1) extracting design information from construction specifications; (2) using the extracted information to match specified material in the construction specification with items from an established database; and (3) retrieving the pricing information of the materials specified in the construction specifications. To test the validity of the proposed method, an experiment was conducted using eight wood construction projects in Detroit, MI. The proposed method was utilized to develop an algorithm that can process the construction specifications to: (1) automatically extract the design information from the construction specifications; and (2) utilize the extracted design information to match materials in a database and retrieve the unit cost of these matched materials. The results from the developed algorithm were compared with the gold standard developed by industry experts. The developed algorithms achieved 99.2% precision and 99.2% recall (i.e., 99.2% F1-measure) for extracted design information instances; 100% precision and 96.5% recall (i.e., 98.2% F1-measure) for extracted materials from the database.

5.1 Background

Building Information Modeling (BIM) provides new techniques to automate the construction processes and enhance the performance of buildings/ infrastructures (Shou et al. 2015; Wong Chong and Zhang 2019; Wong Chong et al. 2020). Nonetheless, while could be automatically obtained from BIM hypothetically; practically, the information obtained from BIM can be limited (Lee et al. 2014). This limited information obtained from BIM can be attributed to the different LOD that can be created in BIM.. An LOD specification provides a reference tool to enhance communications among BIM stakeholders regarding the components and elements in BIMs (BIMForum 2019). There are six primary LOD definitions: LOD 100, LOD 200, LOD 300, LOD 350, LOD 400, and LOD 500 (BIMForum 2017). According to the American Institute of Architects (AIA), models at: (1) LOD 100 are typically developed at the pre-design stage of a project; (2) LOD 200 are typically developed at the schematic stage of the project; (3) LOD 300 are typically developed at the design development stage of the project; (4) LOD 350 are typically developed at the construction documentation stage of the project; (5) LOD 400 are typically developed during the construction stage of a project; and (6) LOD 500 are typically developed during the as-built documentation stage of the project. In generating construction cost estimates, some of the design information needed for cost estimation can be retrieved or extracted from a model at LOD 400 or greater. Due to the dearth of all models at LOD 400 or greater for the purpose of generating cost estimates, estimators continually manually extract these needed design information from design specifications, outline specifications, and construction specifications respectively (Charette and Marshall 1999). According to Nassar (2012), some designs are still illustrated in 2D views. Therefore, at such times, automated information extraction from a thoroughly developed 3D model cannot be easily accomplished. These manual processes of extracting design information from specifications require thorough and in-comprehensive construction knowledge (Staub-French et al. 2003; Ma et al. 2016). Different CBS or WBS are utilized in generating and preparing cost estimates. These CBS and WBS are typically based on the different construction classification systems available .

5.2 Proposed Method

To address the research gap in the automated extraction of design information from construction specifications and automated generation of material lists for construction cost estimation, in this dissertation, a new semantic NLP-based method was proposed for developing an IEM algorithm that extracts design information from construction specifications and matches the extracted design information with materials in a database. The method can be utilized to develop algorithms that: (1) automatically extract the required design information for any building component from an AIA formatted construction specification document; and (2) automatically match the extracted design information with materials in a database. Using the material lists, the method can be further utilized to extract the unit prices of the matched material automatically. The method includes four main steps in developing the IEM algorithm that can be utilized to process construction specifications for extracting design information required for computing cost estimation (Figure 5.1). Step 1: Semantic Modeling, this step defines the hierarchical structure of a building; Step 2: Cost Database Creation, this step classifies the subcomponents/cost items and stores their corresponding unit prices; Step 3: Information Extraction and Matching (IEM) Algorithm Development, this step develops the algorithm for extracting the design instances from construction specifications and matching the extracted design information with the materials in the cost database; Step 4: Database Iteration, this steps loops through the database till a condition is met. The details of these four steps are described in the following subsections, respectively.

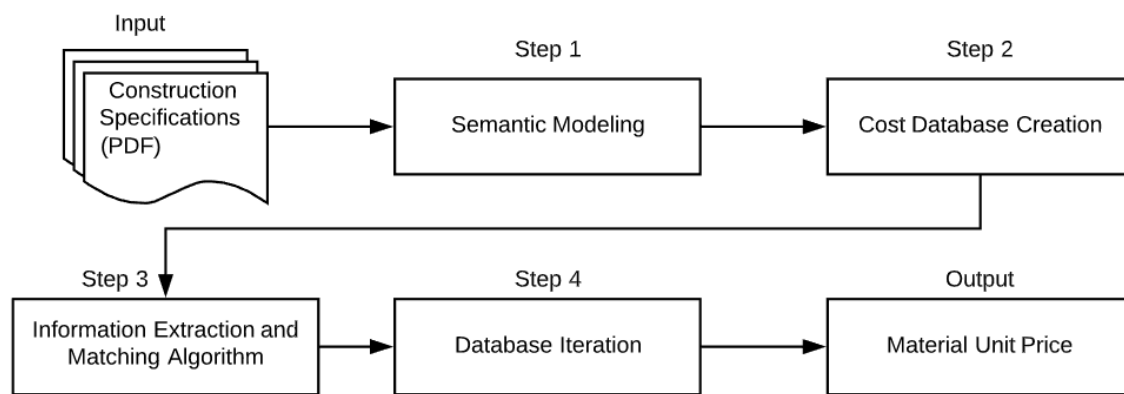


Figure 5.1. Proposed Method

5.2.1 Step 1 - Semantic Modeling

A semantic model in the form of an ontology is used to guide and support the information extraction and matching in an automated fashion. A series of four processes are required for developing the semantic model: (1) creation of the root node; (2) creation of the component nodes (parent node); (3) creation of the sub-component nodes (child nodes); and (4) creation of the properties/attributes of these nodes. In creating the semantic model, a hierarchical structure is followed corresponding to the industry practice of WBS and CBS. A tree data structure is utilized to depict the hierarchical nature of the data. The hierarchical tree model has relationships similar to a “parent-child” relationship. A parent can be related to more than one child, whereas a child can only be related to one parent. A node without a parent is defined as the root node. As an example, in Figure 5.2, *Building Structure* is at the topmost level without any parent and is, therefore, a root node. *Building Structure* acts as a parent to *Roof Component* and *Wall Component*, and in turn, *Wall Component* acts as a parent to *Interior Wall Component* and *Exterior Wall Component*. *Wall Component* as a “child” can only be related to one parent - *Building Structure*. As a “parent,” *Wall Component* is related to more than one component: *Interior Wall Component* and *Exterior Wall Component*. The associations between the parent nodes and the child nodes determine the nodes’ antecedents.

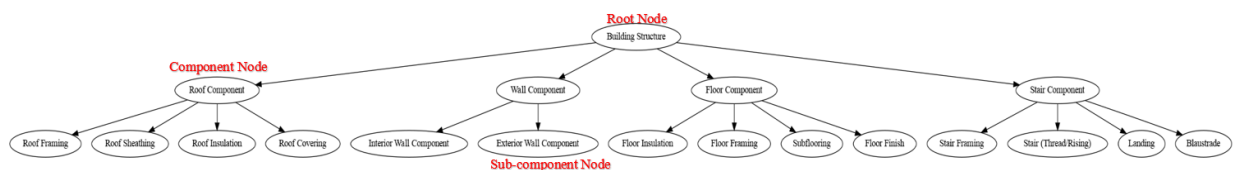


Figure 5.2 Partial Hierarchical Model of a Semantic Model

5.2.2 Step 2- Cost Database Creation

A construction cost database is an automatically indexed directory or table of cost estimating information utilized in generating construction cost estimates. Cost databases are designed to decrease human labor and efforts. The cost database stores and organizes the data created from the semantic modeling step (Step 1). In the created database, there are four main elements (Figure 5.3): (1) identifier – the unique key, a numeric “id” number (1, 2, 3, etc.) that identifies each entry (component line item); (2) building component (referred to as an entity) – the

cost line item from the WBS or CBS (e.g., roof framing); (3) entity type – the description of the entity (e.g., rafter 2 X 8), each database entry represents a unique type of entity; and (4) price – the unit price of each entity type. In Figure 5.3, line item 3, an example is “Building Structure/Roof Component/Roof Framing, Fascia 2X6, 2880.” In this example, the identifier is 3, the building component is *roof framing*, the entity type is *Fascia 2X6*, and the price is 2880 (all prices in the cost database are in U.S. Dollars). In creating the cost database, a series of three processes were designed and used: (1) dense mapping creation, (2) data storing, and (3) component price entering (Figure 6.4). In the cost database, each component of the building (e.g., the wall of a building structure) is represented with an entity type (e.g., *Wall*), its sub-entities, and one or more attributes of each sub-entity. Each attribute, in turn, has more than one value. The entities in the database follow the hierarchical parent-child structure defined in Step 1 (semantic modeling). Figure 5.5 shows that “Wood,” a sub-entity of floor material (finish), has five attributes namely, *Type*, *Thickness*, *Finish*, *Pattern*, and *Grade*. Each of the attributes, in turn, has different values with varying data types. Each attribute is mapped against thickness and another feature depending on the industry nomenclature. For example, “Yellow Pine” is mapped against thickness and then the letter grade values B and C, corresponding to the type of grain.

id	parent	dense_connected_child	price
1	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 1X8	2360
2	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 1X10	2575
3	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 2X6	2880
4	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 2X8	3280
5	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 2X10	3450
6	Node(/Building Structure/Roof Component/Roof Framing)	Fascia - 2X12	3920
7	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 1X8	880
8	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 1X10	980
9	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 2X6	1280
10	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 2X8	1420
11	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 2X10	2025
12	Node(/Building Structure/Roof Component/Roof Framing)	Rafter - 2X12	4025
13	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 1X8	3850
14	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 1X10	2205
15	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 2X6	1980
16	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 2X8	2025
17	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 2X10	2310
18	Node(/Building Structure/Roof Component/Roof Framing)	Ridge Board - 2X12	2450

Figure 5.3. The Cost Database

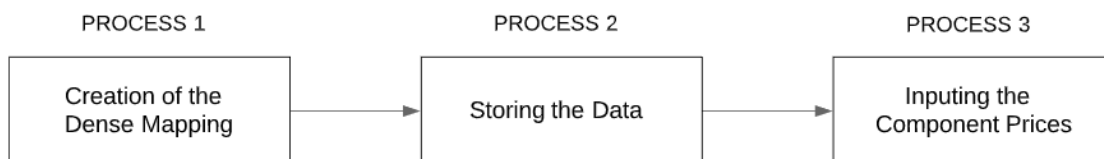


Figure 5.4. Processes involved in the Cost Database Creation

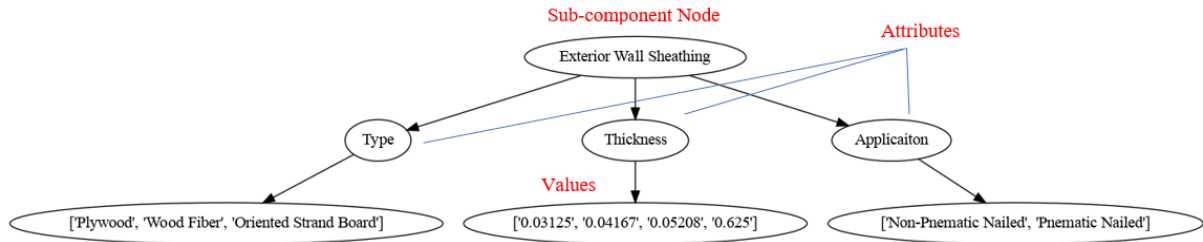


Figure 5.5. Sub-Entity, Attributes & Values (Wood)

5.2.3 Step 3- Information Extraction and Matching Algorithm Development

In developing the Information Extraction and Matching (IEM) algorithm, a series of six processes are defined and used (Figure 5.6). Process I: Data Reading; Process II: Data Preprocessing; Process III: Design Information Identification; Process IV: Semantic Associations; Process V: Information Extraction and Matching (IEM) Algorithm Development; and Process VI: Algorithm Evaluation. Processes 3-6 are iterative in nature; that is, these processes would be repeated till the developed IEM algorithm is robust enough to achieve an acceptable performance. Evaluation data can then be further used as development data in making the algorithm more robust iteratively.

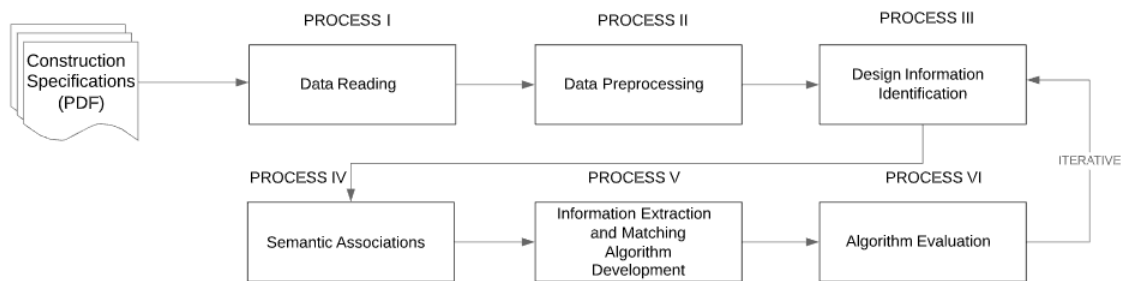


Figure 5.6. Processes Involved in Information Extraction and Matching Algorithm Development

5.2.3.1 Process I- Data Reading

In developing algorithms to extract the design information from construction specifications and match the extracted information with materials from a cost database, the construction specifications data are parsed and analyzed. In parsing and analyzing the construction specifications PDF document, PDF libraries are utilized to: (1) read the data; (2) store the retrieved information; (3) process the page content; and (4) convert the contents into the required format needed for implementation.

5.2.3.2 Process II- Data Preprocessing

In this process, data read from the construction specifications are analyzed and preprocessed. Data preprocessing involves four sub-processes: (1) construction specification data analysis; (2) sentence splitting; (3) tokenization; and (4) morphological analysis. Before extracting the design information from construction specifications, data in the construction specifications are classified, and unnecessary text (for cost estimation) in the construction specifications are filtered out. Construction specifications contain a lot of different types of information ranging from submittals to project execution. As an example, “Part 1 – General” typically contains information related to allowances, submittals, quality assurance, material delivery, material storage, and material handling. This part is filtered out in the proposed method. Data analysis breaks down and decodes the data in the construction specifications to filter out irrelevant texts. Once data analysis is complete and the irrelevant text is filtered, the data is further preprocessed by splitting the relevant text data. Sentence splitting breaks down the complexity of the construction specifications and enables the IEM algorithm to extract all the required information pertaining to each product necessary to determine the unit cost of the material. Construction specifications contain a lot of punctuation marks, and this makes data processing for construction specification challenging. Sentence splitting is used in identifying sentence boundaries and in splitting the data into sentences. The next sub-process after sentence splitting is tokenization. Tokenization breaks a text string into smaller units referred to as tokens. Tokenization can be used to boost the efficiency of a search. For example, the extracted text string “Sheathing - Paper-Surfaced Gypsum Sheathing” breaks into seven tokens through tokenization: ‘Sheathing,’ ‘-,’ ‘Paper,’ ‘-,’ ‘Surfaced,’ ‘Gypsum,’ and ‘Sheathing.’ This sub-process helps boost the robustness of the search by accommodating

different naming conventions in matching building components in the cost database. The last sub-process under data preprocessing is morphological analysis. Morphological analysis is conducted on the tokens to help match the inflectional morphology and derivation morphology of the tokens to their base form word. For example, the extracted string ‘Roof Frame’ would match ‘Roof Framing’ in the database.

5.2.3.3 Process III: Design Information Identification

In this process, the design information to be extracted is identified. To achieve this, a gazetteer comprising the sections and features to be extracted in each section of the specifications is used. For example, weather barrier is a component of the wall assembly that impedes moisture and air passage to the internal spaces through the wall assembly. Weather barriers are typically installed on the exterior face of the building’s envelope, underneath the exterior finish. For proper installation of weather barriers, taping and flashing are required. Taping ensures that the seams of the weather barrier are properly secured, while flashing ensures that the edges around openings (doors and windows) are properly secured. Hence, in the gazetteer, under the “Section 072500 – Weather Barriers,” the feature list includes taping and flashing, and this enables the IE algorithm to extract all information to determine the unit cost of weather barrier.

5.2.3.4 Process IV: Semantic Associations

Semantic associations are the relationships that exist between the connotation of words (Davies and Elder 2004). According to Davies and Elder (2004), there are various types of semantic associations at the word level, including synonymy, hyponymy, and meronymy, among others. Synonymy is the semantic association that exists between words similar in meaning but having different spellings (e.g., gypsum board and drywall); while hyponymy is the semantic association that exists between words such that the meaning of one word incorporates the meaning of other words, e.g., building and structure (Davies and Elder 2004). Meronymy on the other hand, is the semantic association between words that denotes a constituent part of, or a member of something, e.g., cement and concrete (Davies and Elder 2004). In an effort to enhance the robustness of the search strings, the IEM algorithm utilizes these semantic associations and alternative names of the search strings (e.g., building materials).

5.2.3.5 Process V: Information Extraction and Matching Algorithm Development

This process develops the IEM algorithm to extract the features generated in process III (design information identification). Figure 5.7 shows the flow chart of an example IEM algorithm that includes 12 processes and three decisions. Processes 1 - 6 and Decisions 1 – 3 extract the required design instances, while Processes 7 – 10 match the extracted design instances with materials in the database. Processes A – B prompt the user for information if the match decision branches (Decisions 1 to 3) reach a “No” decision. For extracting the design instances, Processes 1 – 3 initiate search strings for division, section, and features based on the labels from the gazetteer. If no matches are found, Processes A – B inform a user that the data (construction specifications) does not contain the required information. If matches are found for all three processes, the algorithm proceeds to Processes 4 - 6 for: (1) extracting the feature token name and attributes; and (2) writing the extracted feature token name and attributes into a search string. Once the feature tokens are extracted and written into a search string, the algorithm proceeds to Processes 7 – 10 for: (1) sentence splitting – identifying sentence boundaries and breaking down the required design data into sentences; (2) tokenization - breaking the search string into tokens where each token is a word, symbol, or punctuation; (3) morphological analysis - matches the variations of a token based on inflectional morphology and derivation morphology to their base form word; (4) semantic associations – matches related terms through different types of semantic associations of the tokens at the word level, including synonymy, hyponymy and meronymy.

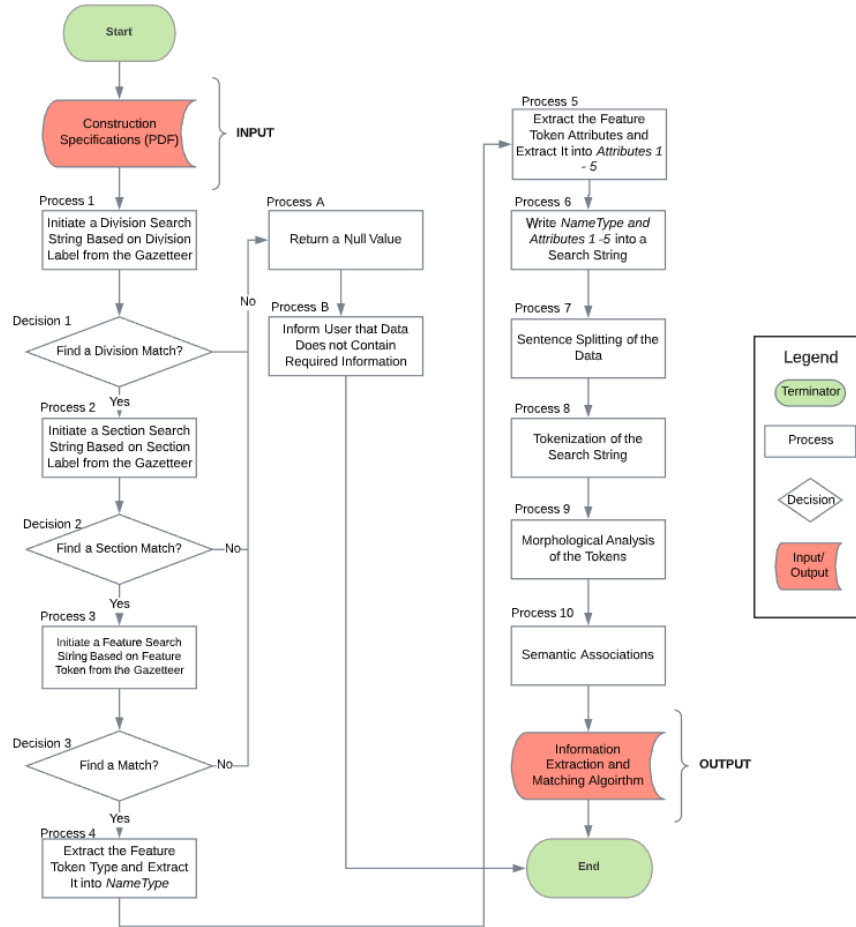


Figure 5.7. Flow Chart of a Sample Information Extraction and Matching Algorithm

5.2.3.6 Process VI: Algorithm Evaluation

In evaluating the robustness of the IEM algorithm, the results obtained using the developed algorithm are compared against the gold standard (results manually generated by industry experts). After each evaluation iteration, if the performance levels of the IEM algorithm are not satisfactory, the data used for evaluation can be further used in the development of the algorithm to improve the robustness of the IEM algorithm.

5.2.4 Step 4: Database Iteration

Iteration is a process where a program performs a set of instructions repeatedly till a condition is met. In our case, each data entity entry in the database (Figure 5.3) is iterated or looped through till a match is found. The stored dictionary for extracted entries comprises entries based

on a “key and value” format. For example, to store information related to sheathing, ‘sheathing’ would represent the key, and any extracted material specification related to sheathing would be the value. A dictionary entry for sheathing could be {‘sheathing’: “[‘Gypsum Board Sheathing’], [‘2/5 inches’]”}. In this example, ‘sheathing’ is the key while “[‘Gypsum Board Sheathing’], [‘2/5 inches’]” is the value. With each dictionary entry, the program parses the key and values from the dictionary to search for a match in the database. Once the database iteration is completed, the algorithm proceeds to extract the material unit price if a match is found. If no match is found, the program displays a null value and prompts the user to add the search string information to the database as a new entry. The program further prompts the user to include a unit price for the saved entry. The program then extracts and prints the value of the entity entry (i.e., material unit price) stored in the database.

5.2.5 Material Unit Price

In the database, costs are stored as unit costs. Each entry represents the cost of one unit of measure of the corresponding material component. In the database, values (the price column of each line item in the database) can be edited to reflect current material cost for the specific geographic location.

5.3 Experimental Testing and Evaluation

To test the effectiveness of the proposed method in this dissertation, it was implemented for extracting the design requirements for eight residential building structures in Detroit, MI. These projects are all wood residential structures comprising two floors, a roof system, stair systems, wall systems, and floor systems. The construction specifications of these buildings were retrieved from local professional architectural firms. The BIM design model shown in Figure 5.8 was developed by the author using a BIM authoring tool, Autodesk Revit 2019. All eight projects are similar in that they are residential building projects and built using wood. Figure 6.8 shows one of the eight projects (Project A). In parsing and analyzing the construction specifications PDF document, the PDFParser, PDFDocument, PDFPageInterpreter, PDFDevice, and PDFResourceManager libraries implemented in python were used (pdfminer, 2017). PDFParser retrieves the data from a PDF file; PDFDocument stores the retrieved data; PDFPageInterpreter is

used to process the page contents; PDFDevice is used to convert the data; PDFResourceManager is used to cache common resources. Three main metrics were used to evaluate the developed IEM algorithm – precision, recall, and F1-measure. Precision is a ratio of correctly matched material list items to the total number of matched material list items. Recall is a ratio of correctly matched material list items to the total number of material list items that should be matched (gold standard). F1-measure is a weighted average of precision and recall measures. In inputting the unit cost of components in the database, the RSMeans Building Construction Costs data (RSMeans 2019) was utilized. MySQL database was utilized for the database, a free, open-source relational database management system. The experiment setup, experimental results, validation results, and analysis are described in the following sub-sections.

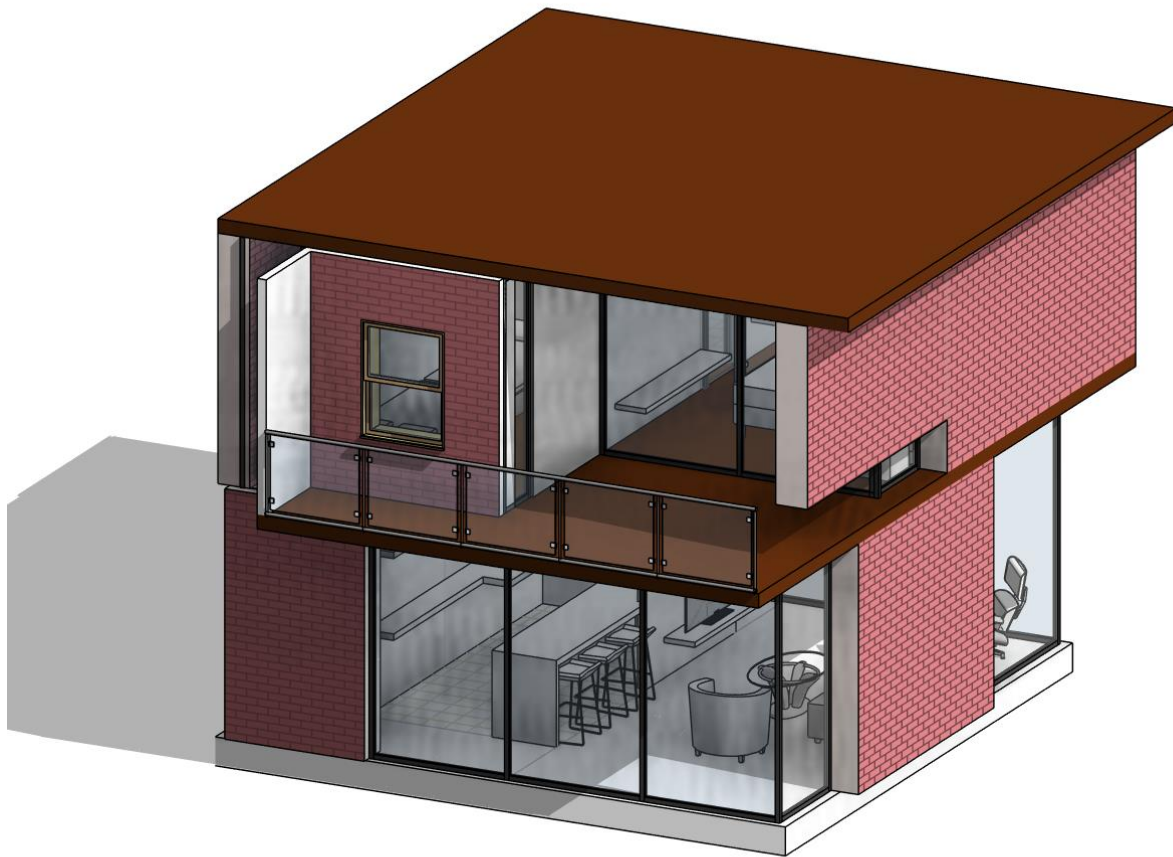


Figure 5.8. Revit Rendering of Project A

5.3.1 Experimental Set-Up and Data

The system's setup composed of a laptop operating on Windows 10 pro; the processor was an Intel ® core TM i7 – 3720 QM CPU @ 2.60 GHz, and the RAM was 16GB. Eight projects (Projects A – H) were collected and used. Projects A and B were utilized for training and development of the algorithms, while Projects C – H were utilized in evaluating the accuracy and robustness of the developed IEM algorithm. After each evaluation iteration, the data used for evaluation were further used in development to improve the robustness of the IEM algorithm.

5.3.1.1 Cost Database

The database is created in Step 2 of the proposed method (Figure 5.1). The database has an adaptable structure that can be adjusted to fit a user's historical cost information. The MySQL workbench (Figure 5.3) interface allows a user to easily store or edit the entry values. MySQL workbench is effective and “can satisfy the most demanding database designers, providing excellent graphical and technological tools” (Letkowski 2014). The chosen database structure is very effective and provides a lot of benefits such as its accuracy, flexibility, and its ability to give multiple users access to the same database. In order to use the generated tree structure, mysql-connector-python library was utilized to save the generated tree structure in the MySQL database. To visualize the saved tree structure in the database, MySQL Workbench 8.022 was utilized. The database (Figure 5.3) contains the root node, component node, sub-component node, dense values, and the price column (RSMeans values). The workbench was used to edit and enter the values in the price column. In extracting the required values from the specifications, the PDF miner library was utilized. The PDF miner library allows for the conversion of the PDF data into text data. After the conversion, regular expressions in python were used to extract the required information from the converted text data. Once the data extraction process has been completed, NLTK was utilized in searching through the database and extracting the price of the required material from the price column of the matched materials (Figure 5.3).

5.3.1.2 Gold Standard

A gold standard is required to perform the evaluation using the precision, recall, and F1-measures. In generating the gold standard, industry experts were consulted to employ the current

industry practices in generating design information and material list. The gold standard containing required design information and material lists was manually built by involving four industry experts (Estimators I, II, III & IV). Each estimator was asked to individually: (1) analyze each project and manually extract the design information for the projects according to a cost summary; and (2) use the manually extracted design information in generating a material list for the cost summaries. To avoid disparities in the classification of components resulting from differences in the standard operating procedures at the different firms, a cost summary given to each estimator was prepared. Each estimator was required to use the cost summary sheet to extract from the specifications a list of MasterFormat section titles and the design information required per section title in generating a material list to support cost estimation computation. For example, under the structural frame, section 061000, Estimator A manually extracted two instances: (1) exterior and load-bearing walls; and (2) 2400fb-2.0E. This indicates that the specified wood (material) for exterior and load-bearing walls is machine stress rated lumber with a grade of 2400fb-2.0E (bending design value of 2,400 psi and stiffness value of 2.0 million psi). All design information extracted (instances) and the corresponding generated material list by the four estimators were tabulated per cost summary item for each project. The highest number of material list items generated per project from the four estimators was chosen as the gold standard.

5.3.1.2.1 Gold Standard - Project C

As described above, in generating the gold standard, each estimator was required to generate a list of design information required per section title, achieving a material list in supporting cost estimation. For example, for exterior wall, three cost components were identified by the estimators to be associated with it: (1) exterior sheathing; (2) air and water barrier; and (3) exterior cladding. The design information for these cost components are identified to be contained in the following construction specification sections: (1) exterior sheathing – construction specification section 061600 (sheathing); (2) air and water barrier – construction specification section 072500 (weather barriers); and (3) exterior cladding – 074646 (fiber-cement siding) and 099113 (exterior painting). The extracted design information from these three construction specification sections are then further used to generate a material list for each project. Table 5.1 tabulates the information elements manually extracted by the estimators (I, II, III & IV) and the number of material list items generated for the exterior wall cost summary for Project C. The

material information items generated by Estimator C was chosen as the gold standard for Project C because Estimator C had the highest number of material items generated. To illustrate this process, the manual extraction process by the estimators for exterior sheathing – a cost component of the exterior wall (construction specification section 061600) is described below.

Table 5.1. Gold Standard – Exterior Wall (Project C)

<i>Cost Summary</i>	<i>Exterior Wall</i>				
<i>Cost Components</i>	Exterior Sheathing	Air and Water Barrier	Exterior Cladding	Total Extracted Design Instances	No of Material List Items
<i>Specification Sections</i>	Section 061600	Section 072500	Section 074646 Section 099113		
<i>Estimator A</i>	8	7	16	31	3
<i>Estimator B</i>	7	6	14	27	3
<i>Estimator C</i> <i>*gold Standard*</i>	9	8	18	35	4
<i>Estimator D</i>	7	6	15	28	3

Exterior Sheathing

Design information regarding exterior sheathing is located in construction specifications section 061600 – sheathing. The estimators identified “Paragraph 2.2 – Wall Sheathing” under “Part 2 – Products” of “Section 061600 – Sheathing” as the paragraph that contained the design information to be extracted. Each estimator highlighted the design information to be extracted. Figure 6.9 shows the highlighted design information extracted by Estimator C. Each highlighted phrase or word is counted as one information element instance. As shown in Figure 5.9, Estimator C highlighted 9 information element instances: (1) Wall Sheathing; (2) Plywood Sheathing; (3) DOC PS 2; (4) Exposure 1; (5) Paper-Surfaced Gypsum Sheathing; (6) water-resistant-treated core; (7) Type and Thickness; (8) Type X; and (9) 5/8 inch thick. The number of instances extracted sometimes varies between the estimators due to subjectivity developed based on individual experience and the estimator’s home firm’s standard operating procedures. As an example, Estimator A extracted 8 instances for exterior sheathing. The estimator did not extract “Exposure 1,” stating that “Exposure 1” plywood sheathing is just a classification for exterior use and therefore not necessary. Estimator B and D both extracted 7 instances each. Both estimators did not extract “water-resistant-treated core,” explaining that the “Type X” classification is

sufficient. Each Estimator further uses the manually extracted instances in generating a material list per cost summary. The number of material list items were counted and tabulated (Table 5.2).

- 2.2 **WALL SHEATHING** ¹
- A. ² Plywood Sheathing: ³ DOC PS 2, ⁴ Exposure 1 sheathing.
 - B. ⁵ Paper-Surfaced Gypsum Sheathing: ASTM C 1396/C 1396M, gypsum sheathing; with water-resistant-treated core and with water-repellent paper bonded to core's face, back, and long edges. ⁶
 - 1. Manufacturers: Subject to compliance with requirements, provide products by one of the following:
 - a. American Gypsum.
 - b. Georgia-Pacific Building Products.
 - c. National Gypsum Company.
 - d. United States Gypsum Company.
 - 2. ⁷ Type and Thickness: ⁸ Type X ⁹ 5/8 inch thick.

Figure 5.9. Design Information Elements Extraction – Estimator C (Project C)

Table 5.2 tabulates: (1) the number of information elements (design requirements) manually extracted for the four cost summary items of Project C - structural frame, exterior wall, interior construction, and roofing & waterproofing; and (2) the number of the material list items generated for Project C.

Table 5.2. Number of Information Elements in Gold Standard Development – Project C

<i>Cost Summary</i>	<i>Structural Frame</i>	<i>Exterior Wall</i>	<i>Interior Construction</i>	<i>Roofing & Waterproofing</i>	<i>Total Instances</i>	
	Information Elements Extracted from Construction Specifications					No. of Material List Items
<i>Estimator A</i>	2	31	62	27	122	16
<i>Estimator B</i>	2	27	65	27	121	17
<i>Estimator C</i> <i>*gold Standard*</i>	2	35	73	31	141	19
<i>Estimator D</i>	2	28	70	25	125	17

5.3.2 Experimental Results

The experiments were conducted under two categories: (1) category 1 – IEM algorithm; and (2) category 2 – unit price extraction. The analysis of the results from the Information Extraction and Matching Algorithm experiment and the Unit Prices Extracted are described below.

Information Extraction and Matching (IEM) Algorithm

Projects A and B were used for the algorithm's development. The construction specifications for Project C was processed with the manuscript's developed IEM algorithm based on Projects A & B. Eleven construction section titles were processed (Table 5.3) by the developed IEM algorithm to automatically extract design information instances and match the extracted instances with materials in the database. For each project, the precision, recall, and F1-measure were measured for the extracted design information instances and the matched materials in the database. Table 5.4 summarizes the experimental results for the extracted design information instances per project, whereas Table 5.5 tabulates the experimental results for the matched materials from the database along with their corresponding performance levels. For Project C, the number of design information instances correctly extracted for structural frame, exterior wall, interior construction, and roofing & waterproofing were 2, 25, 62, and 24 (Table 5.4), respectively, while the number of material list items matched was 16 (Table 5.5). The gold standard contained

2, 35, 73, and 31 instances (Table 5.4) and 19 material list items (Table 5.5). The developed IEM algorithm achieved 94.2% precision, 80.1% recall, and 86.6% F1-measure (Table 5.4) for the extracted design information instances and 88.8% precision, 84.2% recall, and 86.4% F1-measure for the extracted materials from the database (Table 5.5). The relatively low recall for the extracted materials can be associated with the slightly different conventional terms used in construction specifications. This affected the extraction and hence the matching of the material list items. For example, while some architects specified machine stress rated lumber “750f-1.4E, 850f-1.4E, 2400f-2.0E, etc.” as the grades of lumber framing, others might express the same information using “No.1, No. 2, No.3, etc.” Ideally, the goal is to achieve 98% or higher F1-measure (Kumar et al. 2020); therefore, Project C was further utilized as a third development data source to iterate processes III-VI of the proposed method (i.e., Design Information Identification, Semantic Associations, Information Extraction and Matching Algorithm Development, and Algorithm Evaluation). The updated IEM algorithm was further evaluated using Project D. For Project D, the number of design information instances correctly extracted for structural frame, exterior wall, interior construction, and roofing & waterproofing were 2, 23, 59, and 27 (Table 5.4), respectively while the material list items matched were 21 (Table 5.5). The gold standard contained 2, 25, 62, and 30 instances (Table 5.4) and 23 material list items (Table 5.5). The developed IEM algorithm achieved 93.3% recall, 93.3% precision, and 93.3% F1-measure (Tables 5.4) for the extracted design information instances and 91.3% recall, 95.5% precision, and 93.4% F1-measure for the extracted materials from the database (Table 5.5). Although the performance levels increased significantly, it was still far from a 98% F1-measure. Therefore, the iteration continued. After Projects D to G were all utilized to iterate processes III-VI of the proposed method and the updated IEM algorithm evaluated using Project H. The number of design information instances correctly extracted for structural frame, exterior wall, interior construction, and roofing & waterproofing were 2, 29, 65, and 30 (Table 5.4), respectively, while the material list items matched were 28 (Table 5.5). The gold standard contained 2, 29, 64, and 30 design information instances (Table 5.4) and 29 material list items (Table 5.5). The developed IEM algorithm achieved 99.2% recall, 99.2% precision, and 99.2% F1-measure (Tables 5.4) for the extracted design information instances and 96.5% recall, 100.0% precision, and 98.2% F1-measure for the extracted materials from the database (Table 5.5). Figure 5.10 shows the plot of the learning curve for the IEM algorithm

development. As shown in Figure 5.10, the performance increased as the training set size increased. Figure 5.11 shows the partial material list retrieved from Step 4 (Database Iteration).

Table 5.3. Construction Sections Processed (Project C)

Item No	Section Number	Section Title
1	Section 061000	Rough Carpentry
2	Section 061600	Sheathing
3	Section 072100	Thermal Insulation
4	Section 072500	Weather Barriers
5	Section 074646	Fiber-cement Siding
6	Section 092900	Gypsum Board
7	Section 095619	Resilient Tile Flooring
8	Section 096816	Sheet Carpeting
9	Section 097200	Wall Covering
10	Section 099113	Exterior Painting
11	Section 099123	Interior Painting

Table 5.4. Experimental Results

Project	Cost Summary	No. of instances in gold standard	No. of correctly extracted instances	No. of totally extracted instances	Precision	Recall	F1-measure
C	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	35	25	26	96.2%	71.4%	82.0%
	Interior Construction (3)	73	62	68	91.2%	84.9%	87.9%
	Roofing & Waterproofing (4)	31	24	24	100.0%	77.4%	87.3%
	Total/Avg.	141	113	120	94.2%	80.1%	86.6%
D	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	25	23	26	88.5%	92.0%	90.2%
	Interior Construction (3)	62	59	62	95.2%	95.2%	95.2%

Table 5.4 continued

	Roofing & Waterproofing (4)	30	27	29	93.1%	90.0%	91.5%
	Total/Avg.	119	111	119	93.3%	93.3%	93.3%
E	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	25	24	28	85.7%	96.0%	90.6%
	Interior Construction (3)	64	62	68	91.2%	96.9%	94.0%
	Roofing & Waterproofing (4)	32	28	28	100.0%	87.5%	93.3%
	Total/Avg.	123	116	126	92.1%	94.3%	93.2%
F	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	27	26	27	96.3%	96.3%	96.3%
	Interior Construction (3)	65	63	66	95.5%	96.9%	96.2%
	Roofing & Waterproofing (4)	28	27	28	96.4%	96.4%	96.4%
	Total/Avg.	122	118	123	96.7%	95.9%	96.3%
G	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	28	27	28	96.4%	96.4%	96.4%
	Interior Construction (3)	64	64	65	98.5%	100.0%	99.2%
	Roofing & Waterproofing (4)	29	28	29	96.6%	96.6%	96.6%
	Total/Avg.	123	121	124	97.6%	98.4%	98.0%
H	Structural Frame (1)	2	2	2	100.0%	100.0%	100.0%
	Exterior Wall (2)	29	29	29	100.0%	100.0%	100.0%
	Interior Construction (3)	64	64	65	98.5%	100.0%	99.2%
	Roofing & Waterproofing (4)	30	29	30	96.7%	100.0%	98.3%
	Total/Avg.	125	124	124	99.2%	99.2%	99.2%

Table 5.5. Experimental Results

Project	Gold standard chosen	No. of material list items in standard	No. of correctly matched material list items	No. of totally matched material list items	Precision	Recall	F1-measure
C	Estimator C	19	16	18	88.8%	84.2%	86.4%
D	Estimator C	23	21	22	95.5%	91.3%	93.4%
E	Estimator D	20	19	19	100.0%	95.0%	97.4%
F	Estimator C	28	27	27	100.0%	96.4%	98.2%
G	Estimator D	23	22	23	95.6%	95.6%	95.6%
H	Estimator C	29	28	28	100.0%	96.5%	98.2%

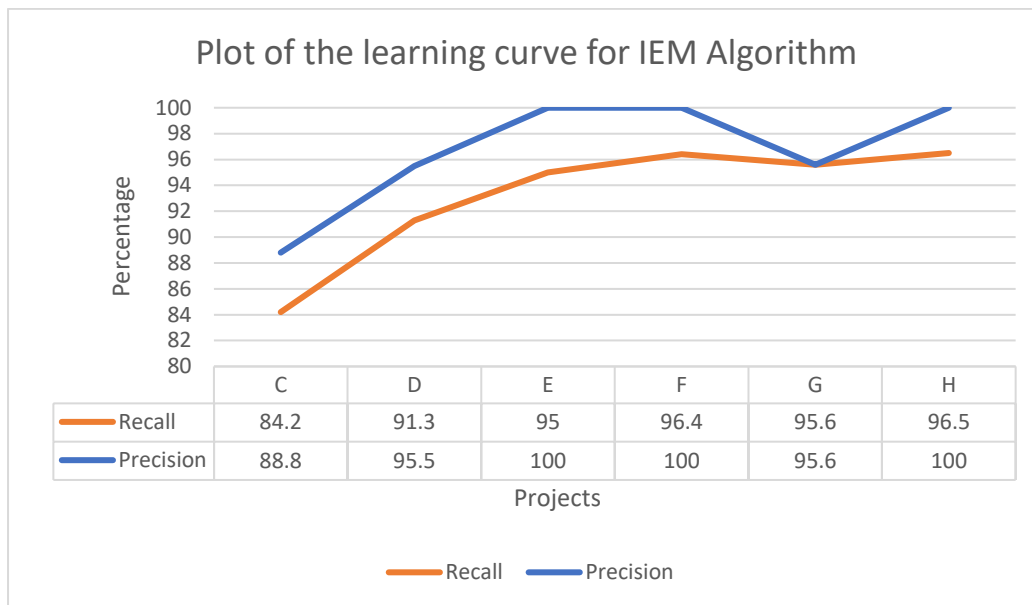


Figure 5.10. Plot of the Learning Curve for IEM Algorithm

2	21	Node('/Building Structure/Roof Component/Roof Sheathing')	Plywood - 0.03125 - Non-Pneumatic Nailed	1.46
3	22	Node('/Building Structure/Roof Component/Roof Sheathing')	Plywood - 0.03125 - Pneumatic Nailed	1.32
28	3487	Node('/Building Structure/Wall Component/Exterior Wall Component/Exterior Wall Sheathing')	Oriented Strand Board - 0.04167 - Non-Pneumatic Nailed	1.55
29	3488	Node('/Building Structure/Wall Component/Exterior Wall Component/Exterior Wall Sheathing')	Oriented Strand Board - 0.04167 - Pneumatic Nailed	1.48
30	3489	Node('/Building Structure/Wall Component/Exterior Wall Component/Exterior Wall Sheathing')	Oriented Strand Board - 0.05208 - Non-Pneumatic Nailed	1.79
31	3490	Node('/Building Structure/Wall Component/Exterior Wall Component/Exterior Wall Sheathing')	Oriented Strand Board - 0.05208 - Pneumatic Nailed	1.62
15	3474	Node('/Building Structure/Wall Component/Exterior Wall Component/Exterior Wall Sheathing')	Plywood - 0.05208 - Pneumatic Nailed	1.79

Partial Material Result List for Sheathing

65	3325	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.04167 - Standard - Taped and Finished (Level 4)	2.01
66	3326	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.04167 - Standard - With Compound Skim Coat (Level 5)	2.46
69	3329	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.04167 - Fire Resistant - With Compound Skim Coat (Level 5)	2.08
70	3330	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.04167 - Water Resistant - No Finish	1.06
79	3339	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.05208 - Standard - No Finish	0.99
80	3340	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.05208 - Standard - Taped and Finished (Level 4)	1.69
81	3341	Node('/Building Structure/Wall Component/Interior Wall Component/Interior Wall Gypsum Board')	0.05208 - Standard - With Compound Skim Coat (Level 5)	2.05

Partial Material Result List for Gypsum Board

Figure 5.11. Partial Material Result List for Sheathing and Gypsum Board

5.4 Conclusions, Contributions, Limitations, and Recommendations for Future Research

5.4.1 Conclusions for the Proposed Extraction Component

There is a gap in developing an automated system for cost estimation that eliminates the subjectivity and human errors in reading construction specifications. To address this gap, a new semantic NLP-based method was proposed in this dissertation for developing an IEM algorithm that extracts design information from construction specifications and matches the extracted design information with materials in a database. The method can be utilized to develop algorithms that: (1) automatically extract the required design information for any building component from an AIA formatted construction specification document; (2) automatically matches the extracted design information with materials in a database and extracts the unit prices of the matched material. The proposed method was validated with eight residential projects in Detroit, MI. The developed IEM algorithm achieved 100.0% precision, 96.5% recall, and 98.2% F1-measure. It was demonstrated

that as the training data increases, the performance levels increase. The developed algorithm utilized 5.56% of the time it took using the current traditional method of extracting design information from construction specifications manually. The performance measures indicate that the proposed method is effective in developing IEM algorithms that extract the design information from construction specifications, match the extracted design information with materials in a cost database and extract the unit prices of the matched materials in supporting construction cost estimation.

5.4.2 Contributions of the Proposed Extraction Component

This part contributes to the body of knowledge in two main ways: First, this study offers a new semantic NLP-based method for developing design information extraction algorithms from construction specifications to support construction cost estimation. In contrast to the state-of-the-art systems that require estimators to extract the needed information from specs for cost estimation manually, the proposed method could reduce the amount of manual effort needed and improve the efficiency and objectivity in specs reading and processing. Secondly, the proposed method can use the extracted design information to match and retrieve material unit prices from a database in a robust manner by addressing variability and ambiguity of item descriptions. This helps automate the cost estimation process in a reliable way and reduces the effort and time required for cost estimation.

5.4.3 Limitations of the Proposed Extraction Component

One main limitation is acknowledged: currently, the developed IEM algorithm was only tested on design information instances for wood elements observed in the development data and construction specifications that followed CSI MasterFormat. In future work, the method will be tested on a broader range of building components (e.g., concrete components, masonry components). Furthermore, the IEM algorithm will be broadened to incorporate other construction specification classification systems such as Unifomat II.

CHAPTER 6. COSTING COMPONENT - AUTOMATED ITEM MATCHING AND PRICING (IMP) FOR WOOD BUILDING ELEMENTS TO SUPPORT BIM-BASED WOOD CONSTRUCTION COST ESTIMATION

A version of this chapter has been published in the in the proceedings of the 2020 ASCE International Workshop on Computing in Civil Engineering by the American Society of Civil Engineers in Tempe, AZ.

Akanbi, T., and Zhang, J., and Lee, Y-C. (2019). “Automated Item Matching and Pricing (IMP) for Wood Building Elements to Support BIM-Based Wood Construction Cost Estimation” *Proc., ASCE Intl. Workshop on Comput. In Civ. Eng.*, ASCE, Reston, VA, 402-410. DOI: 10.1061/97807884482421.051. “With permission from ASCE”

This chapter presents a new method that uses a java constructor and HashMap to create objects, and store and retrieve the created values of the objects. The method utilizes term matching and natural language processing (NLP) techniques to match items from a design model and automatically extract their unit costs from a cost database. The proposed method was tested on estimating a wood construction model retrieved online, and a cost estimate was successfully generated.

6.1 Background

Cost estimation is central to the realization of a successful construction project (Yu et al. 2006). According to Staub-French et al. (2003), one of the fundamental challenges in conducting cost estimation is the expertise required in selecting the appropriate cost parameters, which would ultimately affect the construction cost. Shane et al. (2009) argue that besides complexities in the engineering and construction design of a project, an estimator’s bias can greatly influence the cost of a construction project. In addition to this lack of consistency in the construction cost estimates, the manual cost estimation process is a tedious task subject to human errors (Samphaongoen 2009). Lee et al. (2014) stated that despite the automation of the quantity takeoff (QTO) processes, most

commercial software programs still require estimators to manually match materials of building elements to work items to complete the cost estimation process. Elfaki et al. (2014) reviewed twenty-seven intelligent techniques in cost estimation over a ten-year period and concluded that there are still gaps in the automation of cost estimation despite the existing development, especially in the lack of an intelligent system that addresses the human dependability issues identified from the analysis of these techniques. In recent years, multiple research efforts have been devoted to enable an automated QTO successfully. For example, Akanbi and Zhang (2017) developed a method that automatically reads and extracts quantities of wood building objects by leveraging the fundamental geometric representations of the components in an IFC model. Mandava and Zhang (2016) developed an automated IFC-based QTO method that successfully extracted the needed quantities of bridge components from IFC-based BIMs by leveraging the Cartesian points of the models. Choi et al. (2015) developed a method based on schematic QTO that extracts quantities from BIM architectural elements' data and utilizes ratio formulas to compute the number of materials. However, the matching of building elements with cost items are still mainly performed manually. To address this research gap in matching building elements with cost items, in this dissertation, a new method was proposed for developing automated item matching and pricing (IMP) algorithm using natural language processing (NLP) techniques. The proposed method includes four steps to develop an automated algorithm for IMP, to match building elements from a Building Information Modeling (BIM) design to cost data entries in a cost database. This reduces the need for manual inputs to complete cost estimation processes. The detailed steps are introduced in the next section.

6.2 Proposed Method for Automated Item Matching and Pricing (IMP)

The proposed item matching and pricing (IMP) algorithm development method consist of four steps for cost estimation (Figure 6.1): Step (1) - constructor and HashMap development – define a constructor and its arguments (i.e., parameters) to use in creating new objects, and create a HashMap (i.e., data framework) to store and retrieve values of targeted objects. The created objects represent materials and, therefore, will be referred to as material objects hereafter. For example, “*ProductsCatalogue (material, thickness, cost)*” is a java constructor for *ProductsCatalogue* with three arguments - *material*, *thickness*, and *cost*, which are of *string*, *double*, and *double* data types in Java, respectively. To store the created materials, the *map.put*

method of HashMap is utilized. For example, *map.put (1, material)* – add “material” to the HashMap at Index 1. Step (2) - item matching algorithm development – develop the algorithm for automatically matching items between building objects from the BIM design and cost items from the cost database, and for automatically extracting the unit cost of materials from the cost database for each building component (e.g., wall and floor). The item matching algorithm is developed using natural language processing (NLP) techniques; NLP techniques enable computers to understand and process natural language text (or speech) in a human-like manner (Cherpa 1992). Step (3) - cost estimate computation – the retrieved unit cost from Step (2) is used to compute the cost estimate. Step (4) - evaluation – evaluating the proposed method by comparing a cost estimate based on the developed algorithm with a manually created estimate using existing BIM software. The proposed method is expected to reduce the manual efforts needed to match materials from building design with the appropriate cost components.

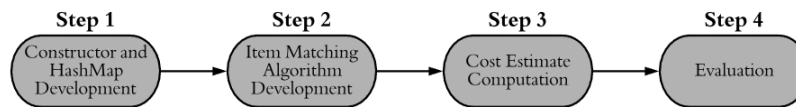


Figure 6.1. Proposed IMP Algorithm Development Method.

6.3 Experimental Testing and Evaluation

The proposed method was tested to estimate the cost of the floor and wall components of a wood structure. The implementation details are described as follows:

6.3.1 Experimental Data

6.3.1.1 Step 1 - Constructor and HashMap Development

In this step, a constructor and a HashMap were developed in Java. The constructor was used to create material objects. The HashMap was used to store or retrieve the newly created objects. A defined constructor has one or more parameters as its arguments. Figure 6.2 shows an example constructor named “*ProductsCatalogue*” that has three parameters - “material,” “thickness,” and “cost.” A HashMap was then used to store the created material objects. In the HashMap, unique identifiers were assigned to each object. When the objects were accessed through the unique identifiers, the values associated with the objects were retrieved. For example,

in Figure 6.2, ('Gypsum Wall Board, 0.05208 ->, \$1.63/sf) is a material object created in "ProductsCatalogue," depicting a "gypsum wall board" with a thickness of 0.05208 (5/8"), the unit cost of which is \$1.63/sf. A HashMap uses the "put" and "get" methods to store or retrieve objects. Each material object was stored by calling the "map.put()" method, and the values were retrieved by calling the "map.getKey()" method.

6.3.1.2 Step 2 - Item Matching Algorithm Development

In this step, an algorithm was developed for automated matching between extracted building elements from BIM and the cost items in the cost database. NLP techniques were used to support the matching, including tokenization and morphological analysis. Tokenization is a process of breaking a piece of text (e.g., the search string) into smaller units (i.e., words, symbols, or punctuations), which are referred to as tokens (Fares et al. 2013). Detailed steps of the developed algorithm are described as follows.

The algorithms include 10 processes (Figure 6.3): *Process 1* initializes a search string (i.e., name of materials) based on material layer information extracted from the IFC. *Process 2* tokenizes the search string from *Process 1*. Tokenization helps enhance the efficiency of a search (Fares et al. 2013). For example, the text string 'Structure, Wood Joist/Rafter Layer' becomes six tokens after the tokenization: 'Structure,' 'Wood,' 'Joist,' '/', 'Rafter,' and 'Layer.' This step helps improve the robustness of accommodating different BIM authoring platforms' proprietary naming conventions of building components. In *Process 3*, morphological analysis is conducted for the tokens to help match all forms of the token with the databases' lowercased names, e.g., if 'Joist,' or 'JOIST' is the token in the search string, the algorithm will execute a search for 'joist.' In *Process 4*, synonym tokens of the search token are generated; creation of synonym tokens ensures that while executing searches for a term, its synonyms are also searched. For example, while searching for 'Joist,' synonym terms such as 'beam' are also searched. *Process 5* uses the resulting terms from *Process 4* to select the appropriate material from the cost database. At this point, a conditional statement *Decision 1* is met.

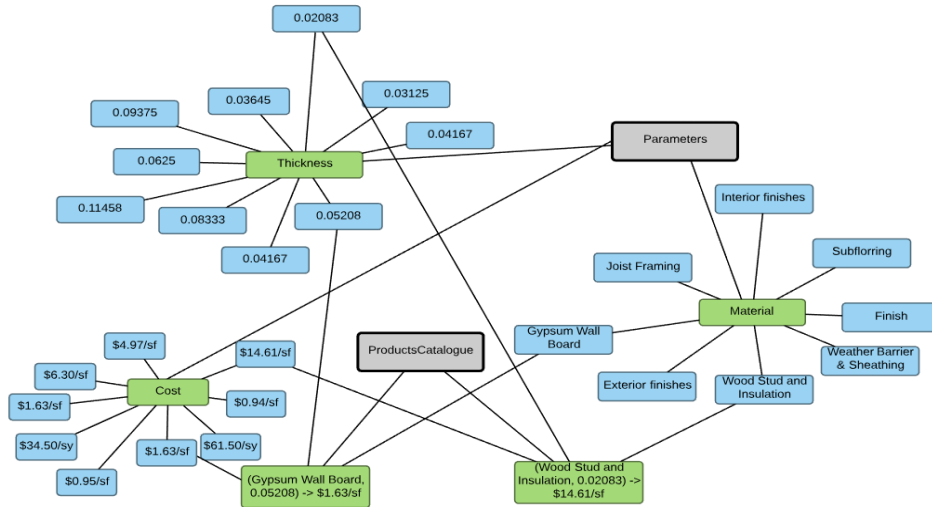


Figure 6.2. Map Structure of a Sample Constructor and HashMap.

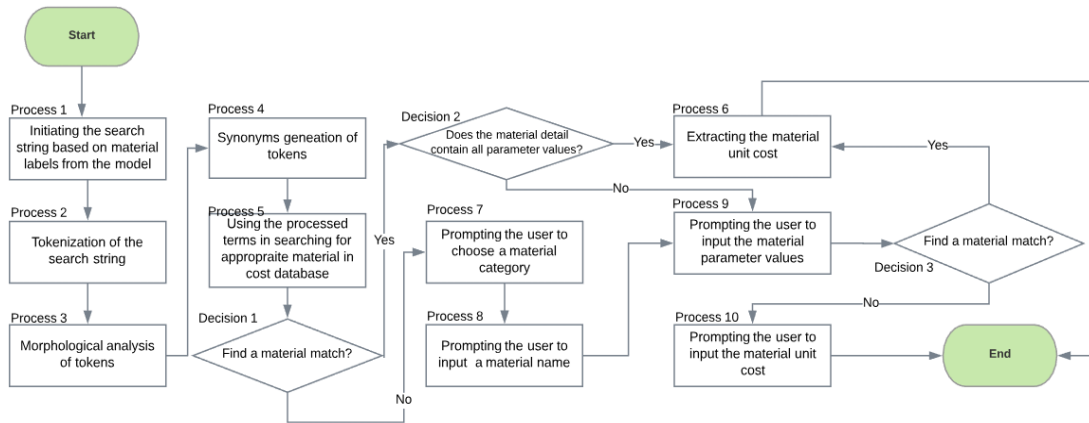


Figure 6.3. Flowchart of the Developed Item Matching Algorithm.

Decision 1 checks if there is a material match found in the cost database. If there is a material match, the algorithm proceeds to a new conditional statement (*Decision 2*); otherwise, it prompts the user for information. *Decision 2* checks if the material detail contains all needed parameter values for picking a unit price. BIM has different LOD specifications. LOD provides a reference that defines the level of details in BIM (Choi et al. 2015). If distinguishing parameter values exist, *Process 6* uses these parameter values to extract the unit cost of the material. For example, the parameter used in selecting the unit cost of a gypsum board is the thickness of the

gypsum board. For wall studs and insulation, the parameters are thickness, height, and spacing. If, however, there is no material match found in the cost database in *Decision 1*, the algorithm proceeds to *Processes 7, 8, and 9*, which prompts the user to input the material category (in the database, materials are categorized based on the component they belong to; e.g., “gypsum wallboard” would be categorized under the wall component), material name, and the material parameters, respectively. Suppose the BIM misses certain material details at *Decision 2*. In that case, the algorithm proceeds to *Process 9* as well to prompt the user to input the material parameter values. At this point, the database is searched again; if a material match is still not found (*Decision 3*), *Process 10* will prompt the user to input the material unit costs manually. However, if sufficient design details exist in the input BIM, none of the manual processes would be activated, and the item matching is fully automated.

6.3.1.3 Step 3 - Cost Estimate Computation

Similar to industry practice in construction cost estimation, the total cost of each wood building element assembly is made up of its cost components (i.e., the materials that make up the assembly grouped for cost purposes). For example, a wall assembly is made up of five cost components following a similar naming convention as in the Industry Foundation Classes (IFC) model: (1) gypsum board, (2) wood stud and insulation, (3) weather barrier & sheathing, (4) exterior finishes, and (5) interior finishes. Whereas the floor components are grouped into three cost components: (1) joist framing, (2) subflooring, and (3) finish.

In the database, costs were stored as unit costs. Each unit cost represents the cost to install one unit of the component (i.e., including all labor, material, and equipment costs). Components may have different units of measures, which dictate the quantity to be multiplied with the unit cost for computing the cost estimate. For example, gypsum board uses a unit of measure of square foot (S.F.), whereas carpet uses a unit of measure of square yard (S.Y.). Therefore, to compute the cost estimates of gypsum board and carpet, the unit cost per S.F. of gypsum board and the unit cost per S.Y. of carpet were used to multiply the corresponding quantities, namely, net area of the wall in S.F. and net area of the floor in S.Y., respectively.

To illustrate these processes in computing the cost estimate, a subcomponent of wall (wood studs and insulation) is used as an example for detailed explanations below.

Figure 6.4. shows an example wall with a “2x4 wood stud,” including details about its subcomponents. The “*MaterialLayerSet*” that follows an IFC naming convention of a typical wall consists of two layers of ‘Gypsum Wall Board,’ one layer of ‘Structure, Wood Joist/Rafter Layer, Batt Insulation,’ and another two layers of ‘Gypsum Wall Board.’ Hence, the following cost variables would be utilized in estimating the costs of the components: (1) unit cost per square foot of gypsum board; (2) unit cost per square foot of wood studs and insulation; and (3) unit cost per square foot of interior finishes.

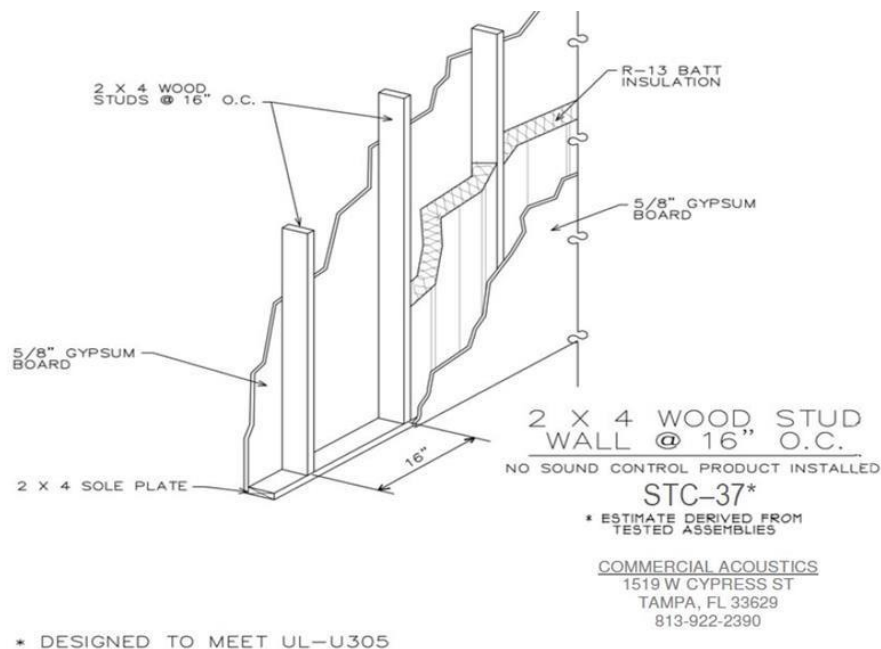


Figure 6.4. Material Layers of a wall (Commercial Acoustics 2017).

6.3.1.3.1 Wood studs and Insulation

Three parameters were used in selecting the unit costs of wood studs and insulation: (1) the thickness of each layer of the material, (2) the height of the wall, and (3) the spacing of the studs. The first and second parameters, the thickness of the material layer, and the height of the wall were extracted during the QTO process. The third parameter, the spacing of the wood studs, could only be retrieved from a LOD 400 BIM and above. In this dissertation, the BIM used was at LOD 300 – the model elements were represented in terms of quantity, size, shape, and orientation within the model. Therefore, in retrieving the spacing of wood studs, *Decision 1* in the developed algorithm (Figure 6.3) did not find a material match. The algorithm then proceeded to *Process 7*,

as illustrated in Figure 6.3. The system prompted the user to choose a material category (category 1- all, category 2 - wall, category 3 - floor). Next, *Process 8* prompted the user to input a material name (wood). Next, *Process 9* prompted the user to input a value (16" O.C. - sixteen inches on center). All other needed parameters were automatically found. At this point, all parameters to retrieve the unit cost of wood studs and insulation had been completed. The systems found a material match (*Decision 3*) from the database. The unit cost of the material was retrieved. The unit of measure for wood studs and insulation was square foot (S.F.). Hence, the retrieved unit cost per square foot of wood studs and insulation coupled with square foot of the area covered by the wood studs and insulation were utilized in computing the cost estimate.

6.3.1.4 Step 4 - Evaluation

A comparison was made between the cost estimate using the method in this dissertation and the cost estimates by a professional estimator based in Detroit, Michigan. The comparison was conducted in two dimensions: (1) estimation results, and (2) needed manual inputs.

5.3.1.4.1 For Estimation Results

There was a 13% difference in cost estimates between the experimental results using the proposed method and prepared by an estimator caused by the different cost data sources used. While the unit cost in the dissertation's database was based on U.S. national averages, the professional estimator's prices were based on their own historical costs data, which were affected by several factors such as availability of material and availability of labor, and labor productivity.

5.3.1.4.2 For Manual Inputs

The processes based on the state-of-the-art commercial software required the estimator to classify cost items manually, whereas the developed IMP method and algorithm extracted the cost items by leveraging the material characteristics of each component in an IFC file, automatically.

```

Run IFCFileParser
"C:\Program Files\Java\jdk1.8.0_111\bin\java" ...
*****
***** Output for FileTesting Data File Wall 3.ifc*****
*****
Height of wall is 8.85826771653528
Length of Wall is 20.7283054461942
Width of Wall is 0.4166666666666668
Area of Wall is: 183.6168789525045
Volume of Wall is: 76.50703289687712
Total Area of Opening: 6.0
Net Area of Wall is : 177.6168789525045
Total Volume of Opening: 2.500000000000002
Net Volume of Wall is : 74.00703289687712
[ IFCMATERIALLAYERSET((#209,#211,#212,#213,#214),'Basic Wall:Interior - 5" Partition (2-hr)');]
'Basic Wall:Interior - 5" Partition 2-hr'
IFCMATERIAL('Gypsum Wall Board');
Thickness of material is 0.0520833333333333
IFCMATERIAL('Gypsum Wall Board');
Thickness of material is 0.0520833333333333
IFCMATERIAL('Structure, Wood Joist/Rafter Layer, Batt Insulation');
Thickness of material is 0.2083333333333333
IFCMATERIAL('Gypsum Wall Board');
Thickness of material is 0.0520833333333333
IFCMATERIAL('Gypsum Wall Board');
Thickness of material is 0.0520833333333333
Unit Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $0.35
Total Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $62.16590763337657
Unit Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $0.35
Total Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $62.16590763337657
Unit Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $0.35
Total Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $62.16590763337657
Unit Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $0.35
Total Cost of Gypsum Wall Board with thickness 0.0520833333333333 is $62.16590763337657
Unit Cost of Structure, Wood Joist/Rafter Layer, Batt Insulation with thickness 0.2083333333333333 is $0.75
Total Cost of Structure, Wood Joist/Rafter Layer, Batt Insulation with thickness 0.2083333333333333 is $133.21265921437836
Total Cost is $381.87628974788464

```

Figure 6.5. Experimental Cost Estimating Results (partial) using the Proposed Method and Corresponding Algorithms

6.4 Conclusions, Contributions, Limitations, and Recommendations for Future Research

6.4.1 Conclusions for the Proposed Costing Component

In this study, an automated item matching and pricing method was developed to reduce manual inputs needed from estimators in BIM-based cost estimation. The proposed method computes the cost estimate by automatically retrieving unit costs from a linked cost database, using an algorithm based on term-based match and natural language processing (NLP) techniques. The proposed method was tested on a wood construction model retrieved online. The experimental results showed that the proposed method successfully computed the cost estimates of the wood components and reduced the need for manual input in matching building components with cost items compared to estimates generated using the state-of-the-art commercial software by a professional estimator. The proposed method provides a foundation for automatically matching

design elements with cost items in a broad range of construction types (e.g., wood, steel, concrete) using IFC-based BIMs.

6.4.2 Contributions of the Proposed Costing Component

This part contributes to the body of knowledge in two main ways: First, the author developed an automated item matching and pricing method to reduce manual inputs needed from estimators. Secondly, the developed method can be used to compute the cost estimate by automatically retrieving unit costs from a linked cost database. The proposed method was tested on a wood construction model retrieved online; the method successfully computed the cost estimates of the wood components while reducing the need of human input.

6.4.3 Limitations of the Proposed Costing Component

One main limitation is acknowledged: the search strings were developed using the known naming conventions of few selected BIM authoring platforms, which may encounter problems when used with unseen BIM authoring platforms. In future work, the item matching algorithms will be expanded to support a more robust search by incorporating a more powerful matching mechanism and more search strings compatible with various BIM authoring tools.

CHAPTER 7. CONCLUSIONS AND RECOMMENDATIONS

The comparison of the IFC-based framework with current state-of-the-art techniques in generating cost estimates (7.2) appears in the following publication:

Akanbi, T., and Zhang, J. (2021). “Framework for Developing IFC-Based 3D Documentation from 2D Bridge Drawings.” *Journal of Computing in Civil Engineering*, submitted.

Akanbi, T., Zhang, J., and Lee, Y-C. (2020). “Data-Driven Reverse Engineering Algorithm Development Method for Developing Interoperable Quantity Takeoff Algorithms Using IFC-Based BIM.” *Journal of Computing in Civil Engineering*, 34(5): 04020036. DOI: 10.1061/(ASCE)CP.1943-5487.0000909. "With permission from ASCE"

Akanbi, T., and Zhang, J., and Lee, Y-C. (2019). “Automated Item Matching and Pricing (IMP) for Wood Building Elements to Support BIM-Based Wood Construction Cost Estimation” *Proc., ASCE Intl. Workshop on Comput. In Civ. Eng.*, ASCE, Reston, VA, 402-410. DOI: 10.1061/9780784482421.051. “With permission from ASCE”

7.1 Summary and Conclusions

The conducted research presented a framework that addresses BIM interoperability in construction to reduce manual efforts in cost estimates’ computations. This dissertation offers a new framework to compute construction cost estimates while reducing interoperability barriers and subjectivity in the computation of cost estimates.

The framework developed in this research leverages: (1) 3D generation methods for semi-automated generation of 3D information models and IFC files from 2D orthographic bridge drawings; (2) a data-driven approach to develop algorithms that take off quantities automatically from IFC-based BIM models; (3) a semantic NLP-based method for developing algorithms that extracts design information from construction specifications and matches the extracted design information with materials in a database; and (4) NLP-based method for the automated generation of cost estimates. The developed framework provides a foundation for developing an intelligent and fully automated construction cost estimation method that improves BIM interoperability and reduces manual efforts in cost estimates’ computations. The framework includes four components

- a modeling component, a quantification component, an extraction component, and a costing component.

The modeling component was utilized in generating 3D information models and IFC output files from 2D plans. The modeling component further converts the developed 3D information models to IFC files which serves as input for the quantification component. The quantification component establishes a new data driven method for extracting the needed quantities from any building object by leveraging the geometric shape representations of the objects in an IFC model. The extraction component was utilized in automatically extracting design information from construction specifications to complement the quantities extracted by the quantification component in generating the cost estimate. The costing component leverages the information generated by both the quantification and extraction components in automatically generating the cost estimates, reducing the manual efforts needed in computing cost estimates.

7.2 Comparison of the IFC-Based Framework with Current State-of-the-art Methods in Generating Cost Estimates

The developed framework was compared with current state-of-the-art techniques in generating cost estimates. Table 7.1 compares the complete steps required by both the state-of-the-art method and the proposed IFC-based framework in computing cost estimates. As shown in table 7.1, the IFC-based method utilized 7.69% of the time it took the state-of-the-art industry practices in generating a cost estimate. This is as a result of the automation of some of the manual processes involved in: (1) generating the 3D model; (2) performing quantity takeoff; (3) extracting needed information from construction specifications; and (4) calculating the cost estimates. Furthermore, the developed method reduced the interoperability barriers by utilizing IFC-based systems and methods in generating the cost estimate.

Table 7.1. Step-by-Step Comparison of the State-of-the-Art Method and the Proposed Framework

Step	State-of-the-Art Method	Gaps/Method	Proposed Framework	Methods
1	Manually extract design information from plans	Manual processes	Import PDF bridge drawings	Manual processes
2	Define terrain in BIM Platform, e.g., road configuration, horizontal alignment, vertical profile, and cross sections.	Manual processes	Convert PDF drawings to PNG files	Automated
3	Create family components – superstructure and substructure	Manual processes	Select required plan sheets(s)	Manual processes
4	Edit each components based on the design information extracted in Step 1	Manual processes	Enter minimum and maximum boundary coordinates	Manual processes
5	Output – 3D information model	Interoperability issues	Output – 3D information model and IFC file	No interoperability issues
6	Import 3D model	Manual processes and Interoperability issues	Import IFC file	Manual processes
8	Assigning model elements to categories	Manual processes	-	-
9	Perform quantity takeoff	Automated	Perform quantity take off	Automated
10	Manually extract design information from construction specifications	Manual processes	Extraction of design information from specifications	Automated
11			Matching of extracted materials to a database	Automated
12	Apply unit cost	Manual processes and interoperability	Retrieving unit costs	Automated
13	Generation of Cost Estimate	Manual	Generation of Cost Estimate	Automated
Avg. Time (hrs.)	6.66		0.512	

Modeling Component (Step 1 – Step 5): Although, a few researchers have proposed methods to convert 2D content to 3D content, there has been considerably less work in the development of systems/methods for geometric shape representation of objects using generative modeling (Girdhar et al. 2016). Fuferi et al. (2010) developed a system that converted 2D vectorial input to 3D pseudo-wireframe. Girdhar et al. (2016) developed a system that maps a 2D image to a 3D voxel grid. The 3D voxel representation from the system proposed by Girdhar et al. (2016) was generative in nature, allowing for the prediction of full 3D voxels of an object from an image. The modeling component developed in this dissertation advances current state-of-the-art generative modeling techniques to develop 3D information models from PDF drawings. Furthermore, there has been considerable amount of effort with regards to the extension of IFC to the infrastructure domain (Bradley et al. 2016). Benning et al. (2017) developed a methodology to enhance the IFC

model for bridges by identifying all missing concepts and classes in an existing IFC bridge model. Huthwohl et al. (2018) developed an IFC-based system to categorize inspection information on RC bridges. Isailovic et al. (2020) developed an IFC-based approach for integrating point-cloud based detection of bridge component damages through a semantic enhancement of the as-built models. The present dissertation advances current knowledge by utilizing the ISO IFC standard to support the conversion of the 3D information model generated from the traditional 2D drawings to IFC output files. In comparison to the current state-of-the-art practices in the industry that require the design/generation of these 3D information models manually, the developed algorithms automated most of manual processes involved in generating these 3D information models.

Quantification Component (Step 6 – 9): Few researchers explored IFC-based QTO methods. 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 differently. BIM applications is still a one-to-one relationship (Lai et al. 2018), 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 proposed quantification component 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. According to buildingSMART (2019), eighty-three BIM software platforms are IFC-certified and therefore compatible with IFC. The quantification component be applied to models built from these BIM platforms together with other BIM/Non-BIM platforms.

Extraction Component and Costing Component (10 -13): In an effort to automate the manual processes involved in construction cost estimation, Staub-French et al. (2003) proposed a feature-driven activity and resource classification system for predicting construction costs by extracting

and matching the activity specifications of a component. Lee et al. (2014) proposed a method utilizing a semantic reasoning mechanism to automate the inference of work conditions by extracting the work conditions and work items from the design information. Mittas et al. (2015) designed a visual tool that compares and computes the accuracy of different cost estimation methods statistically. Choi et al. (2015) developed a method that uses statistical techniques in material ratio calculations to improve the accuracy of schematic cost estimates. Ma et al. (2016) developed an ontology-based approach for formalizing cost specifications in China to support an improved implementation in computer programs. These efforts improved the processes involved in construction cost estimation. However, these efforts are still limited in terms of achieving full automation of construction cost estimation. In contrast, the proposed extraction and costing component: (1) automatically extracted all the cost parameters required for computing the cost estimates from design documents using a major classification system in the construction industry; (2) automatically saved the extracted parameters in a database that can be further utilized for identifying, matching, and retrieving the unit cost of the material from the database; and (3) automated the computation of the cost estimate.

7.3 Research Contributions

This dissertation contributes to the automated generation of cost estimates as follows:

1. Developed a method for the generation of 3D information models from 2D plans and further converts the 3D information models to IFC files.
2. Developed and established a new data-driven method for automated QTO algorithm development using IFC-based BIMs to support construction cost estimation.
3. Developed a new semantic NLP-based method for extracting design information from construction specifications.
4. Developed an automated item matching and pricing method to reduce manual inputs needed from estimators in computing cost estimates.

7.4 Research Limitations

The developed methods and systems in each of the components of the framework has some limitations, including:

1. The research was limited to specific elements of the built environment – bridges and multifamily units.
2. The modeling component algorithms were only tested on reinforced concrete slab beam bridges.
3. The modeling component includes some manual operations.
4. The quantification component algorithms were currently tested on explicitly modeled elements (e.g., walls, slabs), unmodeled elements (e.g., scaffolding) were not tested
5. The extraction component algorithm was only tested on design information instances for wood elements observed in the development data and construction specifications that followed CSI MasterFormat.
6. The costing component search strings were developed using the known naming conventions of few selected BIM authoring platforms, which may encounter problems when used with unseen BIM authoring platforms.

7.5 Recommendations for Future Research

There are a couple of future research work that would reinforce this research. These further directions includes the following:

1. Extension of the modeling component to fully automate the generation of 3D information models and the generation of other construction components such as walls, floors, roofs, and etc.
2. Extension of the quantification component to enable the D-READ method to cover a broader set of IFC data patterns than those observed in the development data, and QTO algorithms for unmodeled elements.
3. Extension of the extraction component to extract design information instances on a broader range of building components (e.g., concrete components, masonry components) and the incorporation of other construction specification classification systems such as Unifomat II.
4. Extension of the item matching algorithms to support a more robust search by incorporating a more powerful matching mechanism and more search strings compatible with various BIM authoring tools.

REFERENCES

- Abanda, F.H., Kamsu-Foguem, B., and Tah, J.H.M. (2017). "BIM - New Rules of Measurement Ontology for Construction Cost Estimation." *Engineering Science and Technology*, 20(2017), 443-459.
- Afsari, K., and Eastman, C. (2016). "A Comparison of Construction Classification Systems Used for Classifying Building Product Models" *Proc., 2016 ASC Annual International Conference*, Denver, CO., 101-108.
- Akanbi, T., and Zhang, J. (2017). "Automated Wood Construction Cost Estimation" *Proc., 2017 ASCE Intl. Workshop on Comput. in Civ. Eng.*, ASCE, Reston, VA., 141-148.
- Akanbi, T., Zhang, J., and Lee, Y-C. (2019). "Automated Item Matching and Pricing (IMP) for Wood Building Elements to Support BIM-Based Wood Construction Cost Estimation" *Proc., 2019 ASCE Intl. Workshop on Comput. in Civ. Eng.*, ASCE, Seattle, WA., 402-409.
- Akanbi, T., and Zhang, J. (2020). "Automated Design Information Extraction from Construction Specifications to Support Wood Construction Cost Estimation" *Proc., 2020 ASCE Intl. Workshop on Comput. in Civ. Eng.*, ASCE, Tempe, AZ., 658-666.
- Akanbi, T., Zhang, J., and Lee, Y-C. (2020). "Data-Driven Reverse Engineering Algorithm Development Method for Developing Interoperable Quantity Takeoff Algorithms Using IFC-Based BIM" *J. Comput. Civ. Eng.*, 34(5): 04020036.
- Akanbi, T., and Zhang, J. (2021). "Framework for Developing IFC-Based 3D Documentation from 2D Bridge Drawings." *Journal of Computing in Civil Engineering*, submitted.
- Akanbi, T., and Zhang, J. (2021). "Design Information Extraction from Construction Specifications to Support Cost Estimation." *Automation in Construction*, submitted.
- Aladag, H., Demirdogen, G., and Isik, Z. (2016). "Building Information Modeling (BIM) use in Turkish Construction Industry." *Procedia Engineering*, 161(2016), 174-179.
- Artifex. (2020). "Ghostscript" < <https://www.ghostscript.com/> > (September 3, 2020).
- Australian National University. (2020). "What is the Difference Between" <<https://asiapacific.anu.edu.au/mapsonline/faq/what-difference-between-png-file-raster-image-and-svg-file-vector-image/>> (October 31, 2020).
- Autodesk. (2017). "Classification Systems and Their Use in Autodesk Revit: Managing the "I" in BIM." <https://www.biminteroperabilitytools.com/classificationmanager/downloads/Autodesk%20Whitepaper%20-%20Classification%20Systems.pdf> > (April 12, 2019).

- Autodesk. (2021). “Revit” <<https://www.autodesk.ca/en/products/revit/new-features?plc=RVT&term=1-YEAR&support=ADVANCED&quantity=1>> (November 10, 2020).
- 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).
- Bae, A., Lee, D., and Park, B. (2016). “Building Information Modeling Utilization for Optimizing Milling Quantity and Hot Mix Asphalt Pavement Overlay Quality” *Canadian Journal of Civil Engineering*, 2016, 43(10): 888-896.
- BIMForum. (2017). “Level of Development Specification Guide: November 2017.” <https://bimforum.org/wp-content/uploads/2017/11/LOD-Spec-2017-Guide_2017-11-06.pdf> (Apr. 10, 2019).
- BIMvision. (2020). “BIMvision 2.24” <<https://bimvision.eu/en/download/>> (September 10, 2020).
- Boukamp, F., and Akinci, B. (2007). “Automated Processing of Construction Specifications to Support Inspection and Quality Control.” *Automation in Construction*, 17(2007), 90-106.
- Brownlee, J. (2019). “A Gentle Introduction to Generative Adversarial Networks (GANs).” <<https://machinelearningmastery.com/what-are-generative-adversarial-networks-gans/>> (September 1, 2020).
- Bucarelli, N., Zhang, J., and Wang, C. (2018). “Maintainability assessment of light design using game simulation, virtual reality and brain sensing technologies.” *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 378-387.
- Charette, R., and Marshall, H. (1999). “Unifomat II Elemental Classification for Building Specifications, Cost Estimating, and Cost Analysis.” National Institute of Standards and technology (NIST).
- Cherpas, C. (1992). “Natural Language Processing, Pragmatics, and Verbal Behavior.” *Analysis of Verbal Behavior*, 10,135-147.
- 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.

- Cheng, J., Lu, Q., and Deng, Y. (2016). "Analytical Review and Evaluation of Civil Information Modeling." *Automation in Construction*, 67(2016), 31-47.
- 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.
- CSI. (1997). SectionFormat: A Recommended Format for Construction Specifications Sections. http://hosting.uaa.alaska.edu/afbeb/AET102/AET102_section_format.pdf> (April 19, 2019).
- CSI. (2012). UniFormat. <http://www.csinet.org/uniformat>> (April 15, 2019).
- Davies, A., and Elder, C. (2004). *The Handbook of Applied Linguistics*. 1st Ed., Blackwell, Massachusetts.
- Zeng, Z., Shi, H., Wu, Y., and Hong, Z. (2015). "Survey of Natural Language Processing Techniques in Bioinformatics." *Computational and Mathematical Methods in Medicine*, 2015.
- Del Pico. (2012). *Estimating Building Costs for the Residential & Light Commercial Construction Professional*. John Wiley & Sons, Inc, New Jersey.
- 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.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.
- 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.
- Eastman, C.O., Teicholz, P., Sacks, R., and Liston, K. (2011). *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd Ed., Wiley, New Jersey.
- Elfaki, A. O., Alatawi, S., and Abushandi, E. (2014). "Using Intelligent Techniques in Construction Project Cost Estimation." *Advances in Civil Engineering*, 2014(107926).
- Fanning, B., Clevenger, C., Ozbek, M., and Mahmoud, H. (2015). "Implementing BIM on Infrastructure: Comparison of the two Bridge Construction Projects." *Practice Periodical on Structural Design and Construction*, 9(2015), 376-384.
- Fares, M., Oopen, S., and Zhang, Y. (2015). "Machine Learning for High-Quality Tokenization Replicating Variable Tokenization Schemes." *Proc., 14th International Conference, CICLing*, Samos, Greece, 231-244.

- 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.
- Furferi, R., Governi, L., Palai, M., and Volpe, Y. (2010). "From 2D Orthographic Views to 3D Pseudo-Wireframe: An Automatic Procedure" *International Journal of Computer Applications*, 2010, 5(6): 18-24.
- Getuli, V., Ventura, M., Capone, P., and Ciribini, A. (2017). "BIM-Based Code Checking for Construction Health and Safety." *Procedia Engineering*, 196(2017), 454-461.
- Hirschberg, J., and Manning, C. (2015). "Advances in Natural Language Processing." *Science*, 349(2015), 261-266.
- Huthwohl, P., Brilakis, I., Borrmann, A., and Sacks, R. (2018). "Integrating RC Bridge Defect Information into BIM Models." *J. Comput. Civ. Eng.*, 2018, 32(3).
- Isailovic, D., Stojanovic, V., Trapp, M., Richter, R., and Hajdin, R. (2020). "Bridge damage: Detection, IFC-Based Semantic Enrichment and Visualization" *Automation in Construction*, 112 (2020) 103088.
- Jorgensen, K. "Classification of Building Object Types – Misconceptions, Challenges and Opportunities." *Proc., 2011 W78 International Conference on Information Technology for Construction*, Sophia Antipolis, France. DOI: itc.scix.net/paper/w78-2011-Paper-24
- Jurafsky, D., and Martin, J. (2009). "*Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition.*" Englewood Cliffs, NJ: Prentice-Hall
- Kaiser, K., and Miksch, S. (2005). "Information Extraction: A survey."
- Kreider, R., Messner, J., and Dubler, C. (2010). "Determining the Frequency and Impact of Applying BIM for Different Purposes on Projects."
<http://bim.psu.edu/uses/FreqBenefit/BIM_Use-2010_Innovation_in_AEC-Kreider_Messner_Dubler.pdf> (Sept. 24, 2017).
- Kumar, B., Cai, H., and Hastak, M. (2017). "An Assessment of Benefits of Using BIM on an Infrastructure Project" *Proc., International Conference on Sustainable Infrastructure.*, New York, NY, 88 - 95.
- Jorgensen, K. "Classification of Building Object Types – Misconceptions, Challenges and Opportunities." *Proc., 2011 W78 International Conference on Information Technology for Construction*, Sophia Antipolis, France.
- Kim, J-U, Kim, Y-J, Ok, H., and Yang, S-H. (2015). "A Study on the Status of Infrastructure BIM and BIM library Development." *Proc., International Conference on Computational Science and Computational Intelligence (CSCI)*, Las Vegas, Nevada, 857-858.

- Kim, T., and Chi, S. (2019). "Accident Case Retrieval and Analyses: Using Natural Language Processing in the Construction Industry." *J. Constr. Eng. Manage.*, 2019, 145(3): 04019004.
- Lau, S., Zakaria, R., Aminudin, E., Saar, C., Yusof, A., and Wahid, C. (2018). "A Review of Application Building Information Modeling (BIM) During Pre-Construction Stage: Retrospective and Future Directions." *Proc., IOP Conf. Series: Earth and Environmental Science (CSCI)*, 143 (2018) 012050
- 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.
- Lu, W., Peng, Y., Shen, Q., and Li, H. (2013). "Generic Model for Measuring Benefits of BIM as a Learning Tool in Construction Tasks." *J. Constr. Eng. Manage.*, 10.1061/(ASCE)CO.1943-782.0000585
- 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.
- Ma, Z., Liu, Z., and Wei, Z. (2016). "A Generic Feature-Driven Activity-Based Cost Estimation process." *Advanced Engineering Informatics*, 17(2003), 96-105.
- Mandava, B., and Zhang, J. (2016). "A New Automated Quantity Takeoff Method for BIM-Based Bridge Designs." *Proc., CIB W78, Conseil International du Bâtiment (CIB)*, Rotterdam, The Netherlands.
- Mastali, M., and Zhang, J. (2017). "Interactive Highway Construction Simulation Using Game Engine and Virtual Reality for Education and Training Purpose." *Proc., 2017 ASCE Intl. Workshop on Comput. in Civ. Eng.*, ASCE, Reston, VA, 399-406.
- Matta, C. (2006). "3D-4D Building Information Modeling." <https://www.gsa.gov/potal/content/105075> (Jun. 2, 2017)

- McGraw-Hill Construction. (2014). "The Business Value of BIM for Construction in Major Global Markets: How Contractors Around the World are Driving Innovation with Building Information Modeling."
https://www.academia.edu/11605146/The_Business_Value_of_BIM_for_Construction_in_Major_Global_Markets_How_Contractors_Around_the_World_Are_Driving_Innovation_With_Building_Information_Modeling (Sept. 27, 2017).
- Mittas, N., Mamalikidis, L., and Angelis, L. (2015). "A Framework for Comparing Multiple Cost Estimation Methods Using an Automated Visualization Toolkit." *Information and Software Technology*, 57(2015), 310-328.
- MODassic 2020. "Vector, Raster, JPG, EPS, PNG – What's the difference?"
<https://modassicmarketing.com/understanding-image-file-types> (October 31, 2020).
- 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.
- Nassar, K. (2012). "Assessing Building Information Modeling Estimating Techniques Using Data from the Classroom." *Journal of Professional Issues in Engineering Education & Practice*, 138(2012), 171-180.
- Nepal, M., Staub-French, S., Potting, R., and Zhang, J. (2013). "Ontology-based Feature Modeling for Construction Information Extraction from a Building Information Model." *J. Comput. Civ. Eng.*, 2013, 27(5): 555-569.
- Wong Chong, O., Baker, C., Afsari, K., Zhang, J. and Roach, M. (2020). "Integration Of BIM Processes In Architectural Design, Structural Analysis, And Detailing: Current Status And Limitations." *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 1203-1212.
- Wong Chong, O., and Zhang, J. (2019). "Game Simulation to Support Construction Automation in Modular Construction Using BIM and Robotics Technology – Stage I." *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 376-383.
- Pauwels, P., Deursen, D., Verstraeten, R., Roo, J., and Meyer, R. (2011). "A Semantic Rule Checking Environment for Building Performance Checking." *Automation in Construction*, 20(2011), 506-518.
- Pdfminer Release 0.0.1 (2017). <https://buildmedia.readthedocs.org/media/pdf/pdfminer-docs/latest/pdfminer-docs.pdf> (November 10, 2019).
- Plebankiewicz, E., Zima, K., and Skibniewski, M. (2015). "Analysis of the First Polish BIM-Based Cost Estimation Application." *Procedia Engineering*, 123(2015), 405-414.
- Polonski, M. (2015). "Application of the Work Breakdown Structure in Determining Cost Buffers in Construction Schedules." *Archives of Civil Engineering*, LXI(2015), pp. 147-161. DOI: 10.1515/ace-2015-0010

- Python. (2020). “Python 3.9.0” <https://www.python.org/downloads/release/python-390/> (November 11, 2020).
- 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.
- Ren, R., and Zhang, J. (2021). “A New Framework to Address BIM Interoperability in the AEC Domain from Technical and Process Dimensions.” *Adv. Civ. Eng.*, 2021, 8824613.
- Ren, R., and Zhang, J. (2020). “Comparison of BIM Interoperability Applications at Different Structural Analysis Stages.” *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 537-545.
- Ren, R., and Zhang, J. (2019). “Model Information Checking to Support Interoperable BIM Usage In Structural Analysis.” *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 361-368.
- Ren, R., Zhang, J., and Dib, H.N. (2018). “BIM interoperability for structural analysis.” *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 470-479.
- RSMeans. (2019). *Building Construction Costs*. 77th Ed., Thinkstock, U.S.A
- Sacks, R., Kaner, I., Eastman, M., and Jeong, Y.S. (2010). “The Rosewood Experiment – Building Information Modeling and Interoperability for Architectural Precast Facades.” *Automation in Construction*, 19(2010), 419-432.
- 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.
- Shane, J., Molenaar, K., Anderson, S., and Schexnayder, C. (2009). “Construction Project Cost Escalation factors.” *Journal of Management in Engineering*, 25(4), 221-229.
- Shou, W., Wang, J., and Wang, X. (2015). “Formalized Representation of Specifications for Construction Cost Estimation by Using Ontology.” *Computer-Aided Civil and Infrastructure Engineering*, 31(2016), 4-17.
- 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 (April 10, 2019).
- Staub-French, S., Fischer, M., Kunz, J., and Paulson, B. (2003). “A Generic Feature-Driven Activity-Based Cost Estimation Process.” *Advanced Engineering Informatics*, 17(2003), 96-105.

- The General Services Administration (GSA). (2017). "PBS-P100: Facilities Standards for the Public Buildings Service."
<https://www.gsa.gov/cdnstatic/2017_Facilities_Standards_%28P100%29%C2%A0.pdf> (October 14, 2019)
- The National Institute of Building Sciences (NIBS). (2007). "National BIM Standard-United States." <<https://buildinginformationmanagement.files.wordpress.com/2011/06/nbimsv1p1.pdf>> (Sept. 23, 2017).
- 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.
- 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. (2021). "New Automated BIM Object Classification Method to Support BIM Interoperability." *J. Comput. Civ. Eng.*, 2019, 33(5):04049033.
- Wu, J., Sadraddin, H., Ren, R., Zhang, J., and Shao, X. (2021). "Invariant Signatures of Architecture, Engineering, and Construction Objects to Support BIM Interoperability between Architectural Design and Structural Analysis." *J. Constr. Eng. Manage.*, 147(1), 04020148.
- Wu, J., and Zhang, J. (2019). "Introducing Geometric Signatures Of Architecture, Engineering, And Construction Objects And A New BIM Dataset." *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 264-271.
- Wu, J., and Zhang, J. (2018). "Automated BIM Object Classification To Support BIM Interoperability." *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 706-715.
- Yu, W., Lai, C., and Lee, W. (2006). "A WICE approach to Real-Time Construction Cost Estimation." *J. Constr. Eng. Manage.*, 147(1): 04020148.
- Zeng, Z., Shi, H., Wu, Y., and Hong, Z. (2015). "Survey of Natural Language Processing Techniques in Bioinformatics." *Computational and Mathematical Methods in Medicine*, 2015(6), pp. 1 -10. DOI: 10.1155/2015/674296.
- Zhang, J., and El-Gohary, N. (2017). "Integrating Semantic NLP and Logic Reasoning into a Unified System for Fully-Automated Code Checking." *Automation in Construction*, 73, 45-57.
- Zhang, J., and El-Gohary, N. (2016). "Semantic NLP-Based Information Extraction from Construction Regulatory Documents for Automated Compliance Checking." *J. Comput. Civ. Eng.*, 2016, 30(2): 04015014.

- Zhang, J., and El-Gohary, N. (2015a). "A Semantic Similarity-Based Method for Semi-Automated IFC Extension." *Proc., ICSC 2015 - The CSCE International Specialty Conference*, CSCE, Montreal, Canada.
- Zhang, J., and El-Gohary, N. (2015b). "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. (2015c). "Automated Information Transformation for Automated Regulatory Compliance Checking in Construction." *J. Comput. in Civ. Eng.*, 29(4), B4015001.
- Zhang, J., and El-Gohary, N. (2014). "Extending Building Information Models Automatically Using Semantic Natural Language Processing Techniques." *Proc., Comput. in Civ. and Build. Eng. (2014)*, ASCE, Reston, VA, 2246-2253.
- Zhang, J., and El-Gohary, N. (2013a). "Information Transformation and Automated Reasoning for Automated Compliance Checking in Construction." *Proc., 2013 ASCE Intl. Workshop on Comput. in Civ. Eng.*, ASCE, Reston, VA, 701-708. Top Three Papers. General Session Presentation.
- Zhang, J., and El-Gohary, N. (2013). "Handling Sentence Complexity in Information Extraction for Automated Compliance Checking in Construction." *Proc., CIB W78 2013*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.
- Zhang, J., and El-Gohary, N. (2012a). "Extraction of Construction Regulatory Requirements from Textual Documents Using Natural Language Processing Techniques." *Proc., 2012 ASCE Intl. Conf. Comput. in Civ. Eng.*, ASCE, Reston, VA, 453-460.
- Zhang, J., and El-Gohary, N. (2012b). "Automated Regulatory Information Extraction from Building Codes Leveraging Syntactic and Semantic Information." *Proc., ASCE Construction Research Congress*, ASCE, Reston, VA, 622-632.
- Zhang, J., and El-Gohary, N. (2011). "Automated Information Extraction from Construction-Related Regulatory Documents for Automated Compliance Checking." *Proc. CIB W78-W102*, Conseil International du Bâtiment (CIB), Rotterdam, The Netherlands.
- Zhang, J., and Laddipeerla, S. (2018). "A feasibility study of IFC-based BIM 4D simulation using commercial systems to support construction planning in the U.S." *Proc., 54th ASC Annual International Conference*, ASC, Fort Collins, CO, 441-448.
- Zhang, F., Fleyeh, H., Wang, X., and Lu, M. (2019). "Construction Site Accident Analysis using Text Mining and Natural Language Processing Techniques." *Automation in Construction.*, 99(2019), 238-248.
- Zhou, P., and El-Gohary, N. (2016). "Automated Extraction of Environmental Requirements from Contract Specifications." *J. Comput. Civ. Eng.*, 2016, 30(2): 04015014

¹APPENDIX A – PERMISSION TO REUSE CONTENT

From: [PERMISSIONS](#)
To: [Temitope Akanbi](#)
Subject: RE: Permission to reuse Conference Proceedings/Journals
Date: Monday, September 28, 2020 12:45:30 PM

Dear Temitope Akanbi,

Thank you for your inquiry. As an original author of an ASCE journal article or proceedings paper, you are permitted to reuse your own content (including figures and tables) for another ASCE or non-ASCE publication (including your thesis), **provided it does not account for more than 25% of the new work.**

A full credit line must be added to the material being reprinted. For reuse in non-ASCE publications, add the words "With permission from ASCE" to your source citation. For Intranet posting, add the following additional notice: "This material may be downloaded for personal use only. Any other use requires prior permission of the American Society of Civil Engineers. This material may be found at [URL/link of abstract in the ASCE Library or Civil Engineering Database]."

For paper #3, "~~Automated design information extraction from construction specifications to support wood construction cost estimation~~", please note in your credit that you are using the **pre-production version of your paper**, with permission from ASCE, and include the book DOI ([10.1061/9780784482889](https://doi.org/10.1061/9780784482889)).

For paper #5, as it is still in the review phase, I cannot provide pre-production permission.

Do note, each license is unique, covering only the terms and conditions specified in it. Even if you have obtained a license for certain ASCE copyrighted content, you will need to obtain another license if you plan to reuse that content outside the terms of the existing license. For example: If you already have a license to reuse a figure in a journal, you still need a new license to use the same figure in a magazine. You need a separate license for each edition.

Sincerely,

Leslie Connelly
Manager, Publications Marketing
American Society of Civil Engineers
1801 Alexander Bell Drive
Reston, VA 20191

PERMISSIONS@asce.org

703-295-6169

Internet: www.asce.org/pubs | www.ascelibrary.org | <http://ascelibrary.org/page/rightsrequests>

A full credit line must be added to the material being reprinted. For reuse in non-ASCE publications, add the words "With permission from ASCE" to your source citation. For Intranet posting, add the following additional notice: "This material may be downloaded for personal use only. Any other use requires prior permission of the American Society of Civil Engineers. This material may be found at [URL/link of abstract in the ASCE Library or Civil Engineering Database]."

To view ASCE Terms and Conditions for Permissions Requests:

Figure A.1

¹ ASCE permission to reuse content in dissertation

2 APPENDIX B – BRIDGE PLANS UTILIZED FOR DEVELOPING IFC-BASED 3D DOCUMENTATION FROM 2D PDF DRAWINGS

INDIANA DEPARTMENT OF TRANSPORTATION

BRIDGE PLANS

FOR SPANS OVER 20 FEET

ROUTE: SR 57 AT: RP 45+94

PROJECT NO. 9999999 P.E. 9999999 R/W 9999999 CONST.

TRAFFIC DATA

ADVISORY	12000
DESIGN	12000
EXISTING	12000
FUNCTIONAL CLASSIFICATION	387
ACCESS CONTROL	

DESIGN DATA

DESIGN SPEED	55
DESIGN LANE WIDTH	12
DESIGN TRUCK	12
DESIGN TRUCK	12

STRUCTURE INFORMATION

STRUCTURE	3 Span, 30' - 40' Over
STATION	1444+50
OVER	Vankle Creek
SPAN	30' - 40' - 30'
SPAN	30' - 40' - 30'

KIN PROJECT INFORMATION

PROJECT	9999999
PROJECT NO.	9999999
PROJECT NO.	9999999
PROJECT NO.	9999999

REQUIRED ELEMENTS:

- Project Information Block (Upper Left)
- Structure Information Table
- Project Numbers
- Reference Point
- Project Location Map
- North Arrow and Scale
- Bridge Design Data Table
- County Location Map
- Latitude and Longitude
- Project Length Table
- Hydrologic Unit Code (Where needed)
- Standard Specification Reference
- Signature Block and P.E. Seal
- Intended Use and Disclaimer Information
- Owner and L&B Employee in Responsible Charge (ECG) signatures

INTENDED USE AND DISCLAIMER INFORMATION:

This set of sample plan sheets is provided for illustrative purposes only. It is not intended to be used as a basis for a contract, nor is it intended to be used as a basis for a contract, nor is it intended to be used as a basis for a contract. The Designer makes no guarantee of the accuracy of data used for this hypothetical design in accordance with the current Indiana Design Manual. The Designer is not responsible for any errors or omissions in this document. In the event of a conflict, the provisions stated in the current Indiana Design Manual and INDOT CAD Standards Manual will govern.

PURPOSE:

The purpose of this drawing is to provide an overview of the project, including project data, design data, project location, and structure information.

INDIANA DEPARTMENT OF TRANSPORTATION

BRIDGE PLANS

FOR SPANS OVER 20 FEET

ROUTE: SR 57 AT: RP 45+94

PROJECT NO. 9999999 P.E. 9999999 R/W 9999999 CONST.

TRAFFIC DATA

ADVISORY	12000
DESIGN	12000
EXISTING	12000
FUNCTIONAL CLASSIFICATION	387
ACCESS CONTROL	

DESIGN DATA

DESIGN SPEED	55
DESIGN LANE WIDTH	12
DESIGN TRUCK	12
DESIGN TRUCK	12

STRUCTURE INFORMATION

STRUCTURE	3 Span, 30' - 40' Over
STATION	1444+50
OVER	Vankle Creek
SPAN	30' - 40' - 30'
SPAN	30' - 40' - 30'

KIN PROJECT INFORMATION

PROJECT	9999999
PROJECT NO.	9999999
PROJECT NO.	9999999
PROJECT NO.	9999999

REQUIRED ELEMENTS:

- Project Information Block (Upper Left)
- Structure Information Table
- Project Numbers
- Reference Point
- Project Location Map
- North Arrow and Scale
- Bridge Design Data Table
- County Location Map
- Latitude and Longitude
- Project Length Table
- Hydrologic Unit Code (Where needed)
- Standard Specification Reference
- Signature Block and P.E. Seal
- Intended Use and Disclaimer Information
- Owner and L&B Employee in Responsible Charge (ECG) signatures

INTENDED USE AND DISCLAIMER INFORMATION:

This set of sample plan sheets is provided for illustrative purposes only. It is not intended to be used as a basis for a contract, nor is it intended to be used as a basis for a contract, nor is it intended to be used as a basis for a contract. The Designer makes no guarantee of the accuracy of data used for this hypothetical design in accordance with the current Indiana Design Manual. The Designer is not responsible for any errors or omissions in this document. In the event of a conflict, the provisions stated in the current Indiana Design Manual and INDOT CAD Standards Manual will govern.

PURPOSE:

The purpose of this drawing is to provide an overview of the project, including project data, design data, project location, and structure information.

INDIANA DEPARTMENT OF TRANSPORTATION

BRIDGE PLANS

FOR SPANS OVER 20 FEET

ROUTE: SR 57 AT: RP 45+94

PROJECT NO. 9999999 P.E. 9999999 R/W 9999999 CONST.

Figure B.1

2 An example of the bridge plans utilized in developing the modeling component

³APPENDIX C – CONSTRUCTION DRAWINGS UTILIZED FOR DEVELOPING D-READ METHOD FOR DEVELOPING INTEROPERABLE QUANTITY TAKEOFF ALGORITHM USING IFC- BASED BIM

2D Arch Drawing files

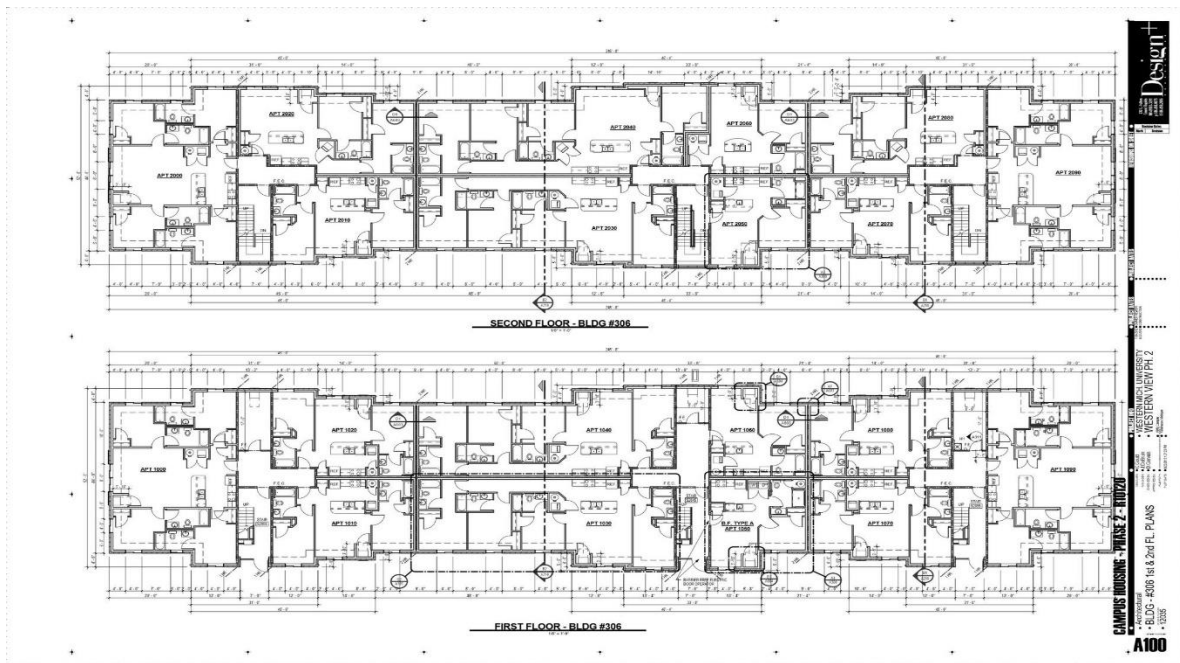


Figure C.1

³ 2D Architectural drawings of the apartment complex utilized in developing the quantification component

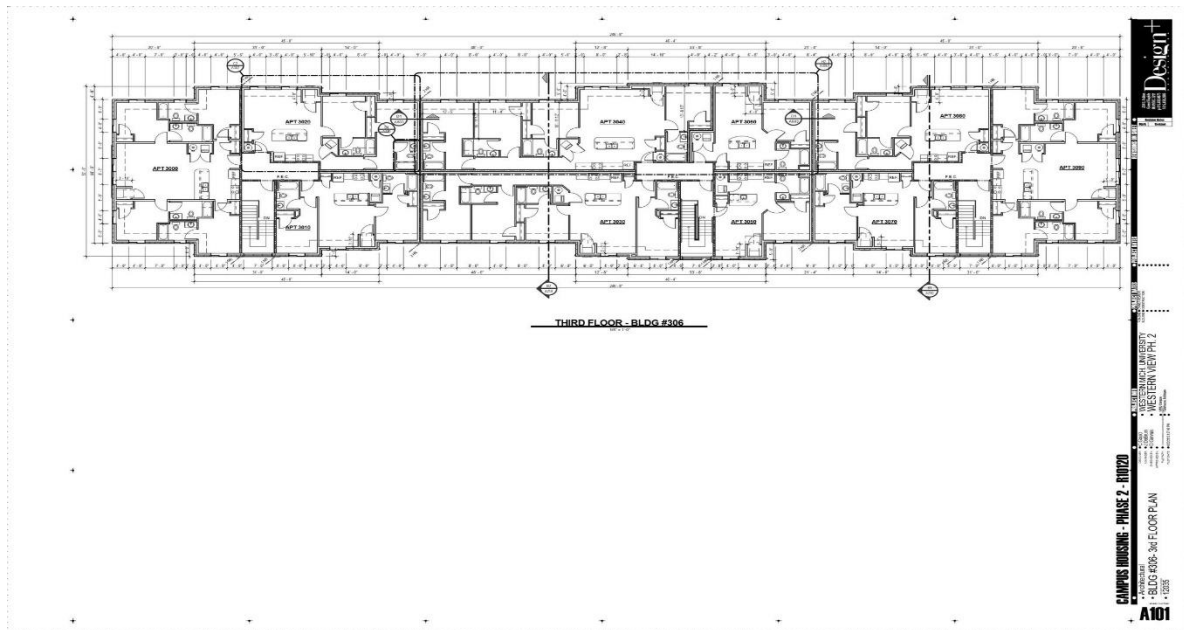


Figure C.2



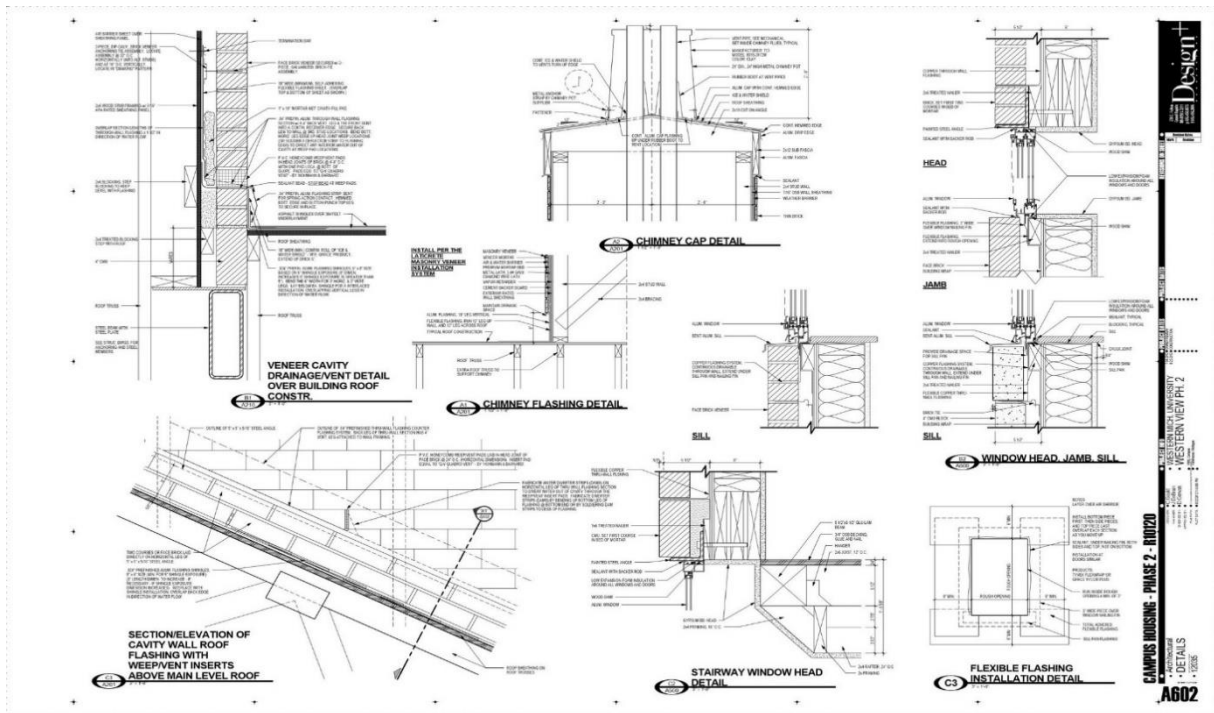


Figure C.4

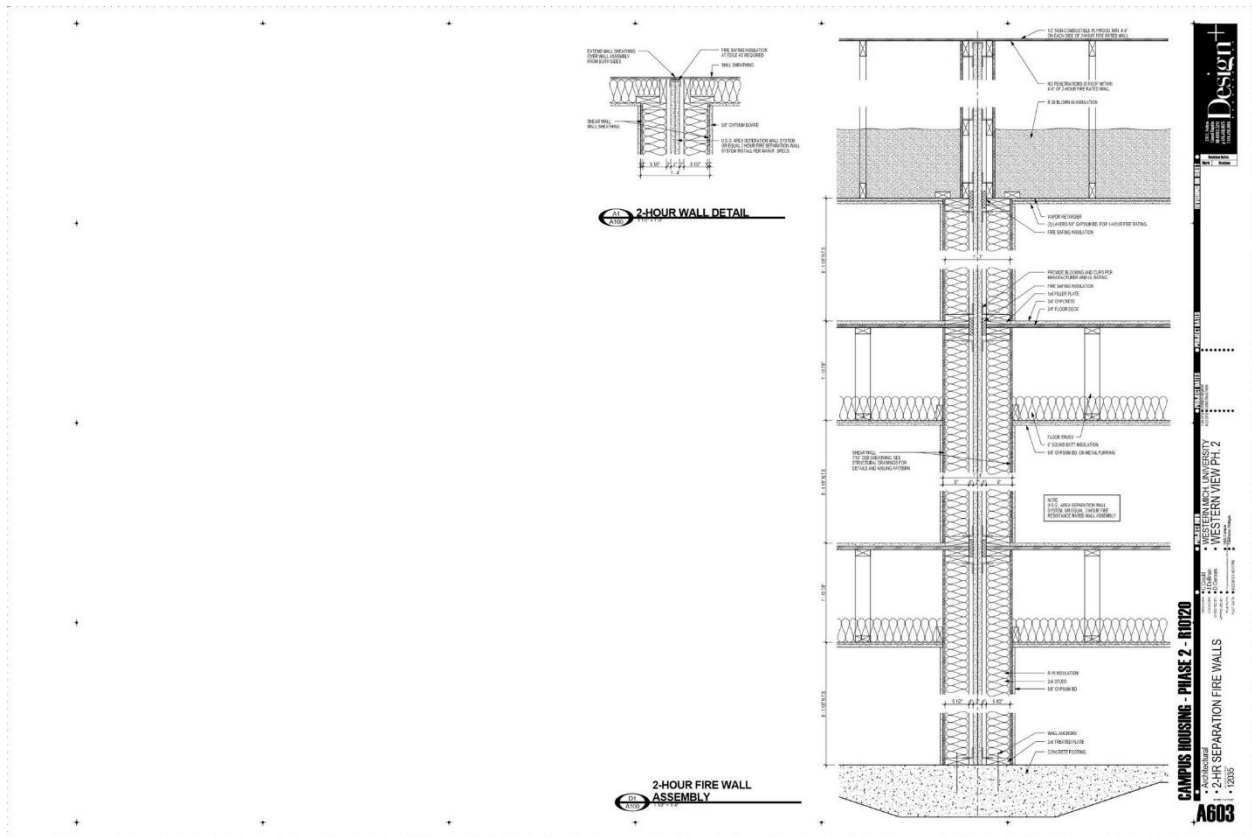
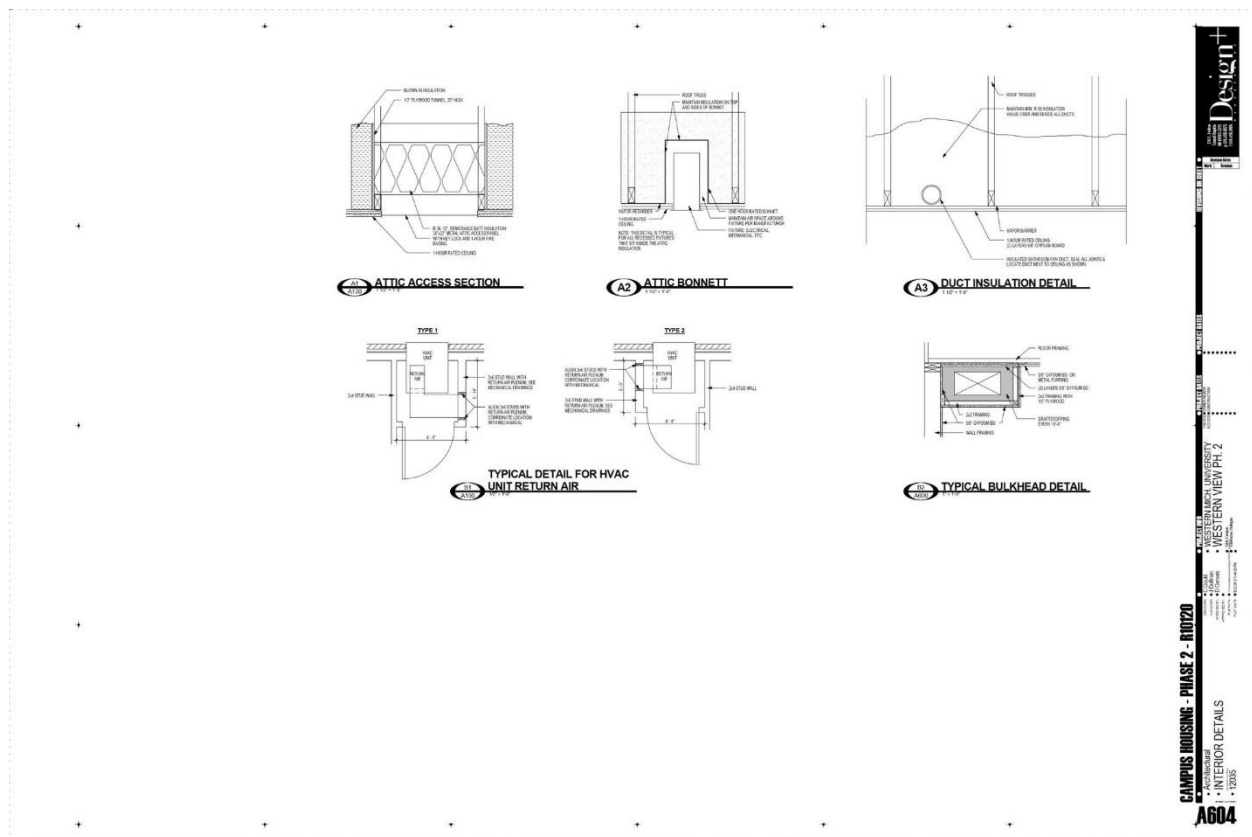


Figure C.5



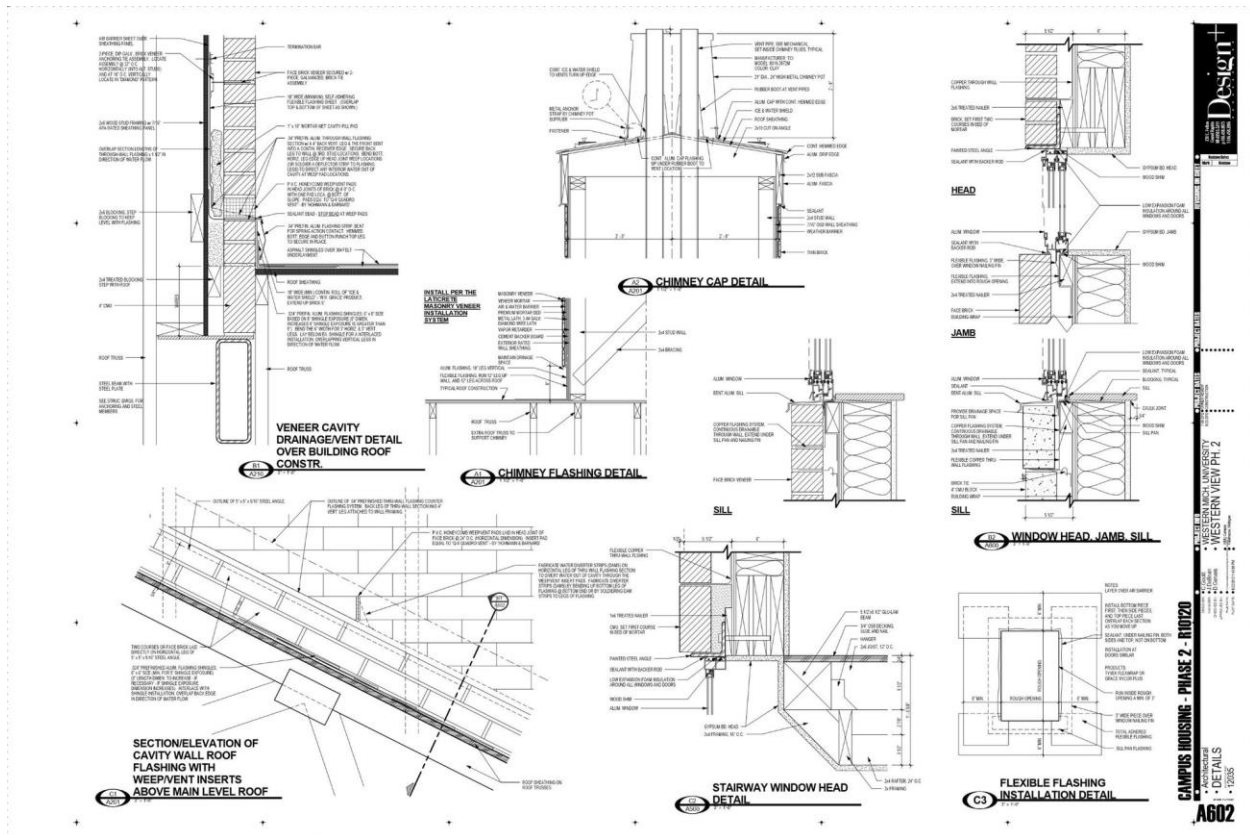


Figure C.7

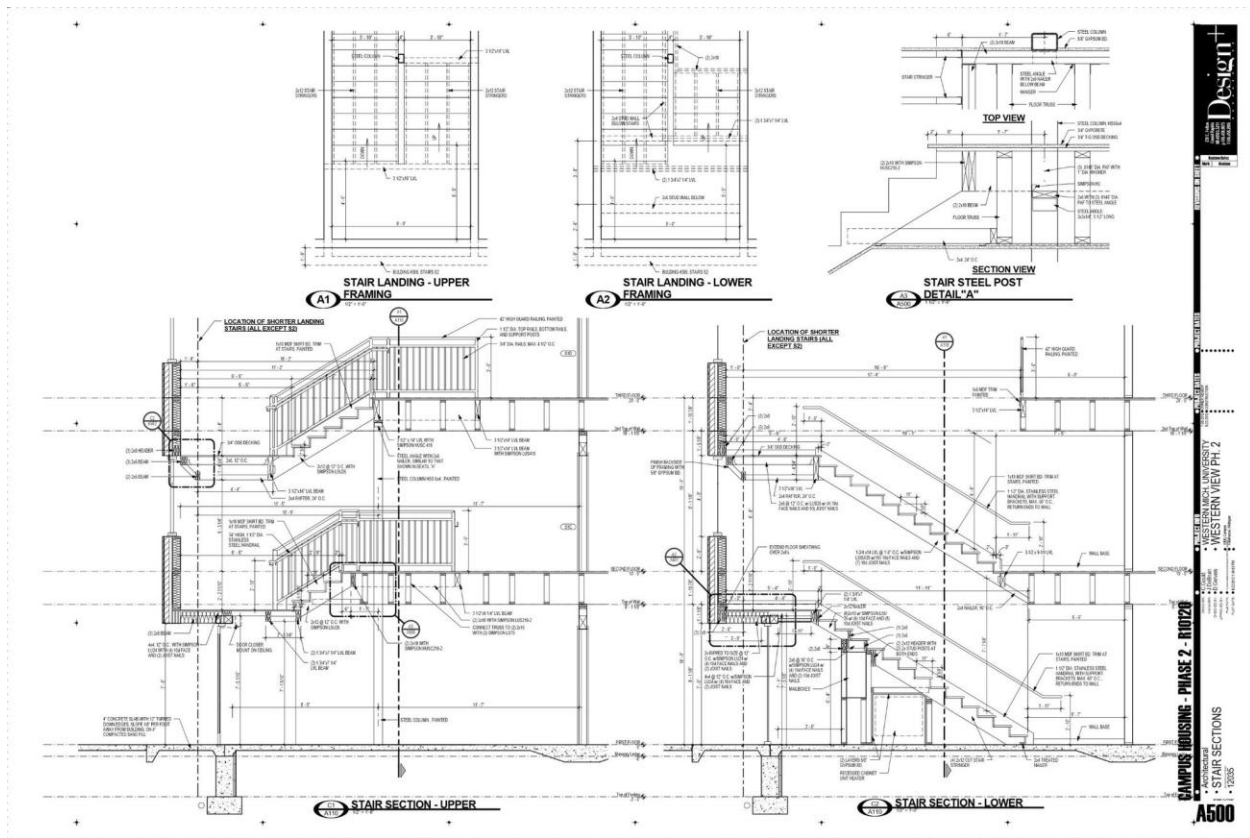


Figure C.8

⁴APPENDIX D – 3D MODEL RENDERING UTILIZED FOR THE QUANTIFICATION COMPONENT



⁴ Front View of the 3D model rendering of one of the building apartment.

⁵APPENDIX E – IFC FILES

```
ISO-10303-21;
HEADER;

/*****
*****
* STEP Physical File produced by: The EXPRESS Data Manager Version
5.02.0100.07 : 28 Aug 2013
* Module: EDMstepFileFactory/EDMstandAlone
* Creation date: Fri Jan 15 09:38:29 2021
* Host: DESKTOP-ETC3MJ7
* Database:
C:\Users\Pixie\AppData\Local\Temp\80806cac-5118-454f-abf4-
cb4d44d118b9\1f9e4916-cb63-41ff-b97b-b1806d9b9aaa\ifc
* Database version: 5507
* Database creation date: Fri Jan 15 09:35:38 2021
* Schema: IFC2X3
* Model: DataRepository.ifc
* Model creation date: Fri Jan 15 09:35:39 2021
* Header model: DataRepository.ifc HeaderModel
* Header model creation date: Fri Jan 15 09:35:39 2021
* EDMuser: sdai-user
* EDMgroup: sdai-group
* License ID and type: 5605 : Permanent license. Expiry
date:
* EDMstepFileFactory options: 020000
*****/
FILE_DESCRIPTION(('ViewDefinition [CoordinationView V2.0]'),'2;1');
FILE_NAME('Project Number','2021-01-15T09:38:29',(''),(''),'The
EXPRESS Data Manager Version 5.02.0100.07 : 28 Aug 2013','21.0.0.383 -
Exporter 21.0.0.383 - Alternate UI 21.0.0.383','');
FILE_SCHEMA(('IFC2X3'));
ENDSEC;

DATA;
#1= IFCORGANIZATION($,'Autodesk Revit 2021 (ENU)',$, $, $);
#5= IFCAPPLICATION(#1,'2021','Autodesk Revit 2021 (ENU)', 'Revit');
#6= IFCARTESIANPOINT((0.,0.,0.));
#9= IFCARTESIANPOINT((0.,0.));
#11= IFCDIRECTION((1.,0.,0.));
#13= IFCDIRECTION((-1.,0.,0.));
#15= IFCDIRECTION((0.,1.,0.));
#17= IFCDIRECTION((0.,-1.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#21= IFCDIRECTION((0.,0.,-1.));
#23= IFCDIRECTION((1.,0.));
#25= IFCDIRECTION((-1.,0.));
#27= IFCDIRECTION((0.,1.));
#29= IFCDIRECTION((0.,-1.));
#31= IFCAXIS2PLACEMENT3D(#6,$,$);
#32= IFCLOCALPLACEMENT(#186,#31);
#35= IFCPERSON($,'','Pixie',$,$,$,$,$);
```

Figure E.1

⁵ IFC Files

```

#37= IFCORGANIZATION($,'','',$,$);
#38= IFCPERSONANDORGANIZATION(#35,#37,$);
#41= IFCOWNERHISTORY(#38,#5,$,.NOCHANGE.,$,$,$,1583291286);
#42= IFCSIUNIT(*,.LENGTHUNIT.,$,.METRE.);
#43= IFCDIMENSIONALEXPONENTS(1,0,0,0,0,0,0);
#44= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.3048),#42);
#45= IFCCONVERSIONBASEDUNIT(#43,.LENGTHUNIT.,$,'FOOT',#44);
#46= IFCSIUNIT(*,.AREAUNIT.,$,.SQUARE_METRE.);
#47= IFCSIUNIT(*,.VOLUMEUNIT.,$,.CUBIC_METRE.);
#48= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,.RADIAN.);
#49= IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#50= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.0174532925199433),#48);
#51= IFCCONVERSIONBASEDUNIT(#49,.PLANEANGLEUNIT.,$,'DEGREE',#50);
#52= IFCSIUNIT(*,.MASSUNIT.,$,.KILO.,$,.GRAM.);
#53= IFCDERIVEDUNITELEMENT(#52,1);
#54= IFCDERIVEDUNITELEMENT(#42,-3);
#55= IFCDERIVEDUNIT((#53,#54),$.MASSDENSITYUNIT.,$);
#57= IFCDERIVEDUNITELEMENT(#42,4);
#58= IFCDERIVEDUNIT((#57),$.MOMENTOFINERTIAUNIT.,$);
#60= IFCSIUNIT(*,.TIMEUNIT.,$,.SECOND.);
#61= IFCSIUNIT(*,.FREQUENCYUNIT.,$,.HERTZ.);
#62= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,.KELVIN.);
#63= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,.DEGREE_CELSIUS.);
#64= IFCDERIVEDUNITELEMENT(#52,1);
#65= IFCDERIVEDUNITELEMENT(#62,-1);
#66= IFCDERIVEDUNITELEMENT(#60,-3);
#67= IFCDERIVEDUNIT((#64,#65,#66),$.THERMALTRANSMITTANCEUNIT.,$);
#69= IFCSIUNIT(*,.LENGTHUNIT.,$,.DECI.,$.METRE.);
#70= IFCDERIVEDUNITELEMENT(#42,3);
#71= IFCDERIVEDUNITELEMENT(#60,-1);
#72= IFCDERIVEDUNIT((#70,#71),$.VOLUMETRICFLOWRATEUNIT.,$);
#74= IFCSIUNIT(*,.ELECTRICCURRENTUNIT.,$,.AMPERE.);
#75= IFCSIUNIT(*,.ELECTRICVOLTAGEUNIT.,$,.VOLT.);
#76= IFCSIUNIT(*,.POWERUNIT.,$,.WATT.);
#77= IFCSIUNIT(*,.FORCEUNIT.,$,.KILO.,$.NEWTON.);
#78= IFCSIUNIT(*,.ILLUMINANCEUNIT.,$,.LUX.);
#79= IFCSIUNIT(*,.LUMINOUSFLUXUNIT.,$,.LUMEN.);
#80= IFCSIUNIT(*,.LUMINOUSINTENSITYUNIT.,$,.CANDELA.);
#81= IFCDERIVEDUNITELEMENT(#52,-1);
#82= IFCDERIVEDUNITELEMENT(#42,-2);
#83= IFCDERIVEDUNITELEMENT(#60,3);
#84= IFCDERIVEDUNITELEMENT(#79,1);
#85= IFCDERIVEDUNIT((#81,#82,#83,#84),$.USERDEFINED.,$,'Luminous
Efficacy');
#87= IFCDERIVEDUNITELEMENT(#42,1);
#88= IFCDERIVEDUNITELEMENT(#60,-1);
#89= IFCDERIVEDUNIT((#87,#88),$.LINEARVELOCITYUNIT.,$);
#91= IFCSIUNIT(*,.PRESSUREUNIT.,$,.PASCAL.);
#92= IFCDERIVEDUNITELEMENT(#42,-2);
#93= IFCDERIVEDUNITELEMENT(#52,1);
#94= IFCDERIVEDUNITELEMENT(#60,-2);
#95= IFCDERIVEDUNIT((#92,#93,#94),$.USERDEFINED.,$,'Friction Loss');

```

Figure E.2


```

#97= IFCDERIVEDUNITELEMENT(#52,1);
#98= IFCDERIVEDUNITELEMENT(#42,1);
#99= IFCDERIVEDUNITELEMENT(#60,-2);
#100= IFCDERIVEDUNITELEMENT(#42,-1);
#101= IFCDERIVEDUNIT((#97,#98,#99,#100),.LINEARFORCEUNIT.,$);
#103= IFCDERIVEDUNITELEMENT(#52,1);
#104= IFCDERIVEDUNITELEMENT(#42,1);
#105= IFCDERIVEDUNITELEMENT(#60,-2);
#106= IFCDERIVEDUNITELEMENT(#42,-2);
#107= IFCDERIVEDUNIT((#103,#104,#105,#106),.PLANARFORCEUNIT.,$);
#109=
IFCUNITASSIGNMENT((#45,#46,#47,#51,#52,#55,#58,#60,#61,#63,#67,#72,#74
,#75,#76,#77,#78,#79,#80,#85,#89,#91,#95,#101,#107));
#111= IFCAXIS2PLACEMENT3D(#6,$,$);
#112= IFCDIRECTION((6.12303176911189E-17,1.));
#114= IFCGEOMETRICREPRESENTATIONCONTEXT($,'Model',3,0.0001,#111,#112);
#117=
IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Axis','Model',*,*,*,*,#114,$,.GR
APH_VIEW.,$);
#119=
IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Body','Model',*,*,*,*,#114,$,.MO
DEL_VIEW.,$);
#120=
IFCGEOMETRICREPRESENTATIONSUBCONTEXT('Box','Model',*,*,*,*,#114,$,.MOD
EL_VIEW.,$);
#121=
IFCGEOMETRICREPRESENTATIONSUBCONTEXT('FootPrint','Model',*,*,*,*,#114,
$,MODEL_VIEW.,$);
#122= IFCPROJECT('20j4G1qv5F3edn5Ngy1suZ',#41,'Project
Number',$,$,'Project Name','Project Status',(#114),#109);
#128= IFCPOSTALADDRESS($,$,$,$,('## Street\X\OD\X\OACity, State
Zip'),$,',','Boston',',','MA'));
#132=
IFCBUILDING('20j4G1qv5F3edn5Ngy1suY',#41,',',,$,$,#32,$,',',.ELEMENT.,$, $
,#128);
#138= IFCCARTESIANPOINT((0.,0.,-3.));
#140= IFCAXIS2PLACEMENT3D(#138,$,$);
#141= IFCLOCALPLACEMENT(#32,#140);
#143= IFCBUILDINGSTOREY('20j4G1qv5F3edn5Nf3_12P',#41,'Top of
Footing',$,$,'Level:8mm Head',#141,$,'Top of Footing',.ELEMENT.,-3.);
#145= IFCCARTESIANPOINT((0.,0.,-0.6666666666666666));
#147= IFCAXIS2PLACEMENT3D(#145,$,$);
#148= IFCLOCALPLACEMENT(#32,#147);
#149= IFCBUILDINGSTOREY('20j4G1qv5F3edn5Nf3_1Cu',#41,'Masonry
Ledge',$,$,'Level:8mm Head',#148,$,'Masonry Ledge',.ELEMENT.,-
0.6666666666666666);
#151= IFCAXIS2PLACEMENT3D(#6,$,$);
#152= IFCLOCALPLACEMENT(#32,#151);
#153= IFCBUILDINGSTOREY('20j4G1qv5F3edn5Nf3_ABS',#41,'FIRST
FLOOR',$,$,'Level:8mm Head',#152,$,'FIRST FLOOR',.ELEMENT.,0.);
#155= IFCCARTESIANPOINT((0.,0.,8.09375));
#157= IFCAXIS2PLACEMENT3D(#155,$,$);

```

Figure E.3

```

ISO-10303-21;
HEADER;

/*****
*****
* STEP Physical File produced by: The EXPRESS Data Manager Version
5.02.0100.07 : 28 Aug 2013
* Module: EDMstepFileFactory/EDMstandAlone
* Creation date: Tue Feb 02 12:01:02 2021
* Host: DESKTOP-FTC3MJ7
* Database:
C:\Users\Pixie\AppData\Local\Temp\e30352fa-80fc-46b8-b8bb-
4bcc25b8fdb0\76438fef-0b22-4458-81e8-d2e97a31db38\ifc
* Database version: 5507
* Database creation date: Tue Feb 02 11:58:03 2021
* Schema: IFC2X3
* Model: DataRepository.ifc
* Model creation date: Tue Feb 02 11:58:04 2021
* Header model: DataRepository.ifc_HeaderModel
* Header model creation date: Tue Feb 02 11:58:04 2021
* EDMuser: sdai-user
* EDMgroup: sdai-group
* License ID and type: 5605 : Permanent license. Expiry
date:
* EDMstepFileFactory options: 020000
*****/
FILE_DESCRIPTION(('ViewDefinition [CoordinationView_V2.0]'),'2;1');
FILE_NAME('Project Number','2021-02-02T12:01:02',(''),(''),'The
EXPRESS Data Manager Version 5.02.0100.07 : 28 Aug 2013','21.0.0.383 -
Exporter 21.0.0.383 - Alternate UI 21.0.0.383','');
FILE_SCHEMA(('IFC2X3'));
ENDSEC;

DATA;
#1= IFCORGANIZATION($,'Autodesk Revit 2021 (ENU)',$, $, $);
#5= IFCAPPLICATION(#1,'2021','Autodesk Revit 2021 (ENU)', 'Revit');
#6= IFCARTESIANPOINT((0.,0.,0.));
#9= IFCARTESIANPOINT((0.,0.));
#11= IFCDIRECTION((1.,0.,0.));
#13= IFCDIRECTION((-1.,0.,0.));
#15= IFCDIRECTION((0.,1.,0.));
#17= IFCDIRECTION((0.,-1.,0.));
#19= IFCDIRECTION((0.,0.,1.));
#21= IFCDIRECTION((0.,0.,-1.));
#23= IFCDIRECTION((1.,0.));
#25= IFCDIRECTION((-1.,0.));
#27= IFCDIRECTION((0.,1.));
#29= IFCDIRECTION((0.,-1.));
#31= IFCAXIS2PLACEMENT3D(#6,$,$);
#32= IFCLOCALPLACEMENT(#157,#31);
#35= IFCPERSON($,'','Pixie', $, $, $, $);

```

Figure E.4

```

#37= IFCORGANIZATION($,'','',$,$);
#38= IFCPERSONANDORGANIZATION(#35,#37,$);
#41= IFCOWNERHISTORY(#38,#5,$,.NOCHANGE.,$,$,$,1583317257);
#42= IFCSIUNIT(*,.LENGTHUNIT.,$.MILLI.,$.METRE.);
#43= IFCSIUNIT(*,.LENGTHUNIT.,$,$.METRE.);
#44= IFCSIUNIT(*,.AREAUNIT.,$,$.SQUARE_METRE.);
#45= IFCSIUNIT(*,.VOLUMEUNIT.,$,$.CUBIC_METRE.);
#46= IFCSIUNIT(*,.PLANEANGLEUNIT.,$,$.RADIAN.);
#47= IFCDIMENSIONALEXPONENTS(0,0,0,0,0,0,0);
#48= IFCMEASUREWITHUNIT(IFCRATIOMEASURE(0.0174532925199433),#46);
#49= IFCCONVERSIONBASEDUNIT(#47,.PLANEANGLEUNIT.,'DEGREE',#48);
#50= IFCSIUNIT(*,.MASSUNIT.,$.KILO.,$.GRAM.);
#51= IFCDERIVEDUNITELEMENT(#50,1);
#52= IFCDERIVEDUNITELEMENT(#43,-3);
#53= IFCDERIVEDUNIT((#51,#52),$.MASSDENSITYUNIT.,$);
#55= IFCDERIVEDUNITELEMENT(#43,4);
#56= IFCDERIVEDUNIT((#55),$.MOMENTOFINERTIAUNIT.,$);
#58= IFCSIUNIT(*,.TIMEUNIT.,$,$.SECOND.);
#59= IFCSIUNIT(*,.FREQUENCYUNIT.,$,$.HERTZ.);
#60= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,$.KELVIN.);
#61= IFCSIUNIT(*,.THERMODYNAMICTEMPERATUREUNIT.,$,$.DEGREE_CELSIUS.);
#62= IFCDERIVEDUNITELEMENT(#50,1);
#63= IFCDERIVEDUNITELEMENT(#60,-1);
#64= IFCDERIVEDUNITELEMENT(#58,-3);
#65= IFCDERIVEDUNIT((#62,#63,#64),$.THERMALTRANSMITTANCEUNIT.,$);
#67= IFCDERIVEDUNITELEMENT(#43,3);
#68= IFCDERIVEDUNITELEMENT(#58,-1);
#69= IFCDERIVEDUNIT((#67,#68),$.VOLUMETRICFLOWRATEUNIT.,$);
#71= IFCSIUNIT(*,.ELECTRICCURRENTUNIT.,$,$.AMPERE.);
#72= IFCSIUNIT(*,.ELECTRICVOLTAGEUNIT.,$,$.VOLT.);
#73= IFCSIUNIT(*,.POWERUNIT.,$,$.WATT.);
#74= IFCSIUNIT(*,.FORCEUNIT.,$,$.NEWTON.);
#75= IFCSIUNIT(*,.ILLUMINANCEUNIT.,$,$.LUX.);
#76= IFCSIUNIT(*,.LUMINOUSFLUXUNIT.,$,$.LUMEN.);
#77= IFCSIUNIT(*,.LUMINOUSINTENSITYUNIT.,$,$.CANDELA.);
#78= IFCDERIVEDUNITELEMENT(#50,-1);
#79= IFCDERIVEDUNITELEMENT(#43,-2);
#80= IFCDERIVEDUNITELEMENT(#58,3);
#81= IFCDERIVEDUNITELEMENT(#76,1);
#82= IFCDERIVEDUNIT((#78,#79,#80,#81),$.USERDEFINED.,'Luminous
Efficacy');
#84= IFCDERIVEDUNITELEMENT(#43,1);
#85= IFCDERIVEDUNITELEMENT(#58,-1);
#86= IFCDERIVEDUNIT((#84,#85),$.LINEARVELOCITYUNIT.,$);
#88= IFCSIUNIT(*,.PRESSUREUNIT.,$,$.PASCAL.);
#89= IFCDERIVEDUNITELEMENT(#43,-2);
#90= IFCDERIVEDUNITELEMENT(#50,1);
#91= IFCDERIVEDUNITELEMENT(#58,-2);
#92= IFCDERIVEDUNIT((#89,#90,#91),$.USERDEFINED.,'Friction Loss');
#94= IFCDERIVEDUNITELEMENT(#50,1);
#95= IFCDERIVEDUNITELEMENT(#43,1);
#96= IFCDERIVEDUNITELEMENT(#58,-2);

```

Figure E.5


```

#257 = IFCFACE((#256));
#258 = IFCPOLYLOOP((#121, #104, #168));
#259 = IFCFACEOUTERBOUND(#258, .T.);
#260 = IFCFACE((#259));
#261 = IFCFACETEDBREP(#76);
#262 = IFCSTYLEDITEM(#261, (#263), $);
#263 = IFCPRESENTATIONSTYLEASSIGNMENT((#264));
#264 = IFCSURFACESTYLE($, .POSITIVE., (#265));
#265 = IFCSURFACESTYLESHADING(#266);
#266 = IFCCOLOURRGB($, 3.450980392156863E-1, 3.96078431372549E-1, 4.470588235294118E-1);
#267 = IFCBUILDINGELEMENTPROXY('1Vd8DCHbf1U88cmnppl6hx', #2, 'IfcBuildingElementProxy -
Gutter:Gutter:170885', 'Description of Object', $, #268, #273, $, $);
#268 = IFCLOCALPLACEMENT(#38, #269);
#269 = IFCAXIS2PLACEMENT3D(#270, #271, #272);
#270 = IFCCARTESIANPOINT((0., 0., 0.));
#271 = IFCDIRECTION((0., 0., 1.));
#272 = IFCDIRECTION((1., 0., 0.));
#273 = IFCPRODUCTDEFINITIONSHAPE($, $, (#295));
#274 = IFCPROPERTYSET('0$YmfUfqN5UwKMv52k6LPT', #2, 'SU_InstanceSet', $, (#275, #276));
#275 = IFCPROPERTYSINGLEVALUE('Owner', $, IFCLABEL(""), $);
#276 = IFCPROPERTYSINGLEVALUE('Status', $, IFCLABEL(""), $);
#277 = IFCRELDEFINESBYPROPERTIES('38gL4xXsjOigrAC7iU3Zqu', #2, $, $, (#267), #274);
#278 = IFCPROPERTYSET('2mleqF9_z1bgANN8BLAIHk', #2, 'SU_DefinitionSet', $, (#279, #280, #281));
#279 = IFCPROPERTYSINGLEVALUE('Price', $, IFCLABEL(""), $);
#280 = IFCPROPERTYSINGLEVALUE('Size', $, IFCLABEL(""), $);
#281 = IFCPROPERTYSINGLEVALUE('Url', $, IFCLABEL(""), $);
#282 = IFCRELDEFINESBYPROPERTIES('2BXSKNG2v71f7YC0k8GaUD', #2, $, $, (#267), #278);
#283 = IFCPROPERTYSET('2C7tQbZO97T9GEkgEKMaVi', #2, 'CompositionType', $, (#284));
#284 = IFCPROPERTYSINGLEVALUE('IfcElementCompositionEnum', $, IFCLABEL('complex'), $);

```

Figure E.7

```

#285 = IFCREDEFINESBYPROPERTIES('0XJeek4BTBH8Ro$qHC5eh6', #2, $, $, (#267), #283);
#286 = IFCPROPERTYSET('3zSAbxbULANvPYqGA1rCMZ', #2, 'ObjectType', $, (#287));
#287 = IFCPROPERTYSINGLEVALUE('IfcLabel', $, IFCLABEL('Gutter:Gutter:41283'), $);
#288 = IFCREDEFINESBYPROPERTIES('0PBmVkJ$EnCTOHAZpnexVqs', #2, $, $, (#267), #286);
#289 = IFCPROPERTYSET('2Pn5WrWuD3uBuYu$ksfSH', #2, 'Name', $, (#290));
#290 = IFCPROPERTYSINGLEVALUE('IfcLabel', $, IFCLABEL('Gutter:Gutter:170885'), $);
#291 = IFCREDEFINESBYPROPERTIES('1maC54l0r33v3vTTWHdG7X', #2, $, $, (#267), #289);
#292 = IFCPROPERTYSET('2Ka8ZsKSD5j96PMe5nUXaO', #2, 'Pset_DistributionFlowElementCommon', $,
(#293));
#293 = IFCPROPERTYSINGLEVALUE('Reference', $, IFCLABEL('Gutter'), $);
#294 = IFCREDEFINESBYPROPERTIES('1QxQga9X5AtO2kGRMMhmO$', #2, $, $, (#267), #292);
#295 = IFCSHAPEREPRESENTATION(#20, 'Body', 'Brep', (#481));
#296 = IFCCLOSEDSHELL((#302, #308, #312, #316, #320, #324, #328, #332, #336, #339, #343, #346,
#352, #356, #360, #364, #368, #372, #376, #380, #384, #388, #392, #396, #399, #402, #405, #408, #411,
#414, #417, #420, #423, #426, #429, #432, #435, #438, #441, #444, #447, #450, #453, #456, #459, #462,
#465, #468, #471, #474, #477, #480));
#297 = IFCPOLYLOOP((#298, #299, #300));
#298 = IFCCARTESIANPOINT((-1746.734059718618, 6441.282639232679, 8738.95136186538));
#299 = IFCCARTESIANPOINT((-1743.559058127175, 6441.282639232679, 8738.95136186538));
#300 = IFCCARTESIANPOINT((-1746.734059718618, 6441.282639232679, 8617.214131296256));
#301 = IFCFACEOUTERBOUND(#297, .T.);
#302 = IFCFACE((#301));
#303 = IFCPOLYLOOP((#304, #305, #306));
#304 = IFCCARTESIANPOINT((-1868.559058127174, 6441.282639232679, 8726.251362950185));
#305 = IFCCARTESIANPOINT((-1865.38405653573, 6441.282639232679, 8723.076361358741));
#306 = IFCCARTESIANPOINT((-1868.559058127174, 6441.282639232679, 8713.551360309699));
#307 = IFCFACEOUTERBOUND(#303, .T.);
#308 = IFCFACE((#307));
#309 = IFCPOLYLOOP((#310, #305, #304));
#310 = IFCCARTESIANPOINT((-1855.859059211979, 6441.282639232679, 8726.251362950185));

```

Figure E.8

⁶APPENDIX F – CONSTRUCTION DRAWINGS UTILIZED FOR AUTOMATED DESIGN INFORMATION EXTRACTION FROM SPECIFICATIONS

ANNEX BUILDING SPECIFICATIONS

SPECIFICATIONS	
Project manual and contract documents for	SET #:

The Fields at Southpointe

Oxford, Ohio

Club House and Apartments

December 18th, 2015

Architects signature and seal:

KJG Architecture, Incorporated
527 Sagamore Parkway W, Suite 101
West Lafayette, IN 47906

KJG Architecture, Incorporated

527 Sagamore Parkway W, Suite 101
West Lafayette, Indiana 47906

Office: 765.497.4598 Email: kelly@kjgarchitecture.com Fax: 765.497.4599

Figure F.1

⁶ An example of the construction specifications utilized in developing the extracting component

⁷APPENDIX G – SOURCE CODE- QUANTIFICATION COMPONENT

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Jan 27 14:54:00 2020

@author: Temitope Akanbi
"""

import os
import pandas as pd
import re

# PLEASE CHANGE THE FILENAME HERE:
filename = r"C:\Users\Pixie\PycharmProjects\AUTOCON002\Testing Data File Wall 1.1fc"

mapped_data = {}

wall, floor, stairs = False, False, False

with open(filename, 'r') as file:
    data = file.readlines()
    for i, line in enumerate(data[27:]):
        mapped_data[line.split("=")[0]] = line.split("=")[-1]

wall_data, roof_data, stairs_data, floor_data, material_data = {}, {}, {}, {}, {}

for key, value in mapped_data.items():
    if "TFCWALLSTANDARDCASE" in value:
```

⁷ Source code for the quantification component


```

C = 1
lookup_key = []
for key, value in mapped_data.items():
    if "IFCWALLSTANDARDCASE" in value:
        new_key = value.split(",")[6]
        # print(new_key)
        lookup_key.append(new_key)

if lookup_key:
    definition_shape = []
    for key in lookup_key:
        definition_shape.append(mapped_data[key].split(",")[-1][:4])

if definition_shape:
    shape_data = []
    for key in definition_shape:
        if "SweptSolid" in mapped_data[key]:
            shape = mapped_data[key].split(",")[-1].split(",")[0].replace('(', ')')
            shape = shape.replace(')', ').replace('; ', ').replace("\n", "")
            # print(shape)
            shape_data.append(shape)

depth = []
if shape_data:
    profile_def = []
    for key in shape_data:
        p_d = mapped_data[key].split(",")[0]
        p_d = re.findall(r'^d+', p_d)
        p_d = '#' + p_d[0]
        profile_def.append(p_d)

```

```

        dep = mapped_data[key].split(" ")[-1]
        dep = dep.replace("'", "").replace(";", "").replace("\n", "")
        depth.append(dep)
        # print(dep)

Xdim, Ydim = [], []
if profile_def:
    outer_curve_id = []
    for key in profile_def:
        if "IFCRECTANGLEPROFILEDEF" in mapped_data[key]:
            x_dim = mapped_data[key].split(" ")[-2]
            y_dim = mapped_data[key].split(" ")[-1].replace("'", "").replace(";", "").replace("\n", "")
            Xdim.append(x_dim)
            Ydim.append(y_dim)
        elif "IFCARBITRARYCLOSEDPROFILEDEF" in mapped_data[key]:
            oc_id = mapped_data[key].split(" ")[2].replace("'", "").replace(";", "").replace("\n", "")
            outer_curve_id.append(oc_id)

# PROCESS 6.1
trim_one, trim_two = [], []
if outer_curve_id:
    wall_segment_one = []
    for key in outer_curve_id:
        w_segment = mapped_data[key].split(" ")[0]
        w_segment = re.findall(r'^d+', w_segment)
        w_segment = '#' + w_segment[0]
        wall_segment_one.append(w_segment)
        # print(w_segment)

parent_curve = []

```

```

for key in wall_segment_one:
    p_curve = mapped_data[key].split(",")[2].replace("'", "").replace(";", "").replace("\n", ")
    parent_curve.append(p_curve)

for key in parent_curve:
    t_one = mapped_data[key].split(",")[1]
    t_two = mapped_data[key].split(",")[2]
    t_one = re.findall(r"\d*\.\d+|\d+", t_one)[0]
    t_two = re.findall(r"\d*\.\d+|\d+", t_two)[0]
    trim_one.append(t_one)
    trim_two.append(t_two)

radius = []
for key, value in mapped_data.items():
    if "IFCCIRCLE(" in value:
        r_data = value.split(",")[1]
        r_data = re.findall(r"\d*\.\d+|\d+", r_data)[0]
        radius.append(r_data)

# width_of_wall = radius[1] - radius[2]
length_of_wall = []
if trim_one:
    for i in range(len(trim_one)):
        wall_l = 2 * (3.14) * C * (1 - ((trim_one[i] - trim_two[i]) / 360))
        length_of_wall.append(wall_l)

# PROCESS 11.1
openingRepresentation = []
for key, value in mapped_data.items():
    if "IFCOPENINGELEMENT" in value:

```

```

        # print(value)
        value = value.split(",")[6]
        # print(value)
        openingRepresentation.append(value)

openingDefShape = []
if openingRepresentation:
    for key in openingRepresentation:
        od_shape = mapped_data[key].split(",")[2]
        od_shape = od_shape.replace("(", "").replace(")", "").replace(";", "").replace("\n", "")
        openingDefShape.append(od_shape)

openingItem = []
if openingDefShape:
    for key in openingDefShape:
        o_item = mapped_data[key].split(",")[3]
        o_item = o_item.replace("(", "").replace(")", "").replace(";", "").replace("\n", "")
        openingItem.append(o_item)

openingArea, openingWidth = [], []
if openingItem:
    for key in openingItem:
        area = mapped_data[key].split(",")[0]
        area = re.findall(r'd+', area)
        area = '#' + area[0]
        openingArea.append(area)
        width = mapped_data[key].split(",")[3]
        width = width.replace("(", "").replace(")", "").replace(";", "").replace("\n", "")
        openingWidth.append(width)

```

```

XOdim, YOdim = [], []
if openingArea:
    try:
        for key in openingArea:
            xo_dim = mapped_data[key].split(",")[3]
            yo_dim = mapped_data[key].split(",")[4]
            yo_dim = yo_dim.replace("'", "").replace(";", "").replace("\n", "")
            XOdim.append(xo_dim)
            YOdim.append(yo_dim)
    except:
        pass

```

```

wall_data['depth'] = depth
wall_data['Xdim'] = XOdim
wall_data['Ydim'] = YOdim
wall_data['trim_one'] = trim_one
wall_data['trim_two'] = trim_two
wall_data['radius'] = radius
wall_data['openingWidth'] = openingWidth
wall_data['xodim'] = XOdim
wall_data['yodim'] = YOdim
wall_data['length_of_wall'] = length_of_wall

```

```

elif "IFCRELDEFINESBYPROPERTIES" in value:
    stair_properties = []
    properties = value.split(",")[5]
    properties = properties.replace("'", "").replace(";", "").replace("\n", "")
    stair_properties.append(properties)

```

```

property_value = []
if stair_properties:
    for key in stair_properties:
        try:
            p_value = value[value.find("(")+2:value.find(")"]].split(",")
            p_value_1 = mapped_data[key].split(",")[4]
            if "(" in p_value_1 and ")" in p_value_1:
                p_value_1 = p_value_1.replace("(", "").replace(")", "").replace(";", "").replace("\n", "")
                property_value.append(p_value_1)
            else:
                p_value_2 = mapped_data[key].split(",")[5]
                p_value_3 = mapped_data[key].split(",")[6]
                p_value_4 = mapped_data[key].split(",")[7]
                property_value.append(p_value_2)
                property_value.append(p_value_3)
                property_value.append(p_value_4)
        except Exception as e:
            pass

RiserNumber, RiserHeight, RequiredHeadroom, ThreadNumber, ThreadLength = [], [], [], [], []
if property_value:
    for i in property_value:
        p_single_value = mapped_data[i]
        if "numberofriser" in p_single_value.lower():
            r_number = p_single_value.split(",")[2]
            r_number = re.findall(r"^\d*\d+\d+", r_number)
            RiserNumber.append(r_number)
        elif "riserheight" in p_single_value.lower():

```

```

        r_height = p_single_value.split(",")[2]
        r_height = re.findall(r"\d*\.\d+\.\d+", r_height)
        RiserHeight.append(r_height)
    elif "requiredheadroom" in p_single_value.lower():
        r_headroom = p_single_value.split(",")[2]
        r_headroom = re.findall(r"\d*\.\d+\.\d+", r_headroom)
        RequiredHeadroom.append(r_headroom)
    elif "numberofthread" in p_single_value.lower():
        t_number = p_single_value.split(",")[2]
        t_number = re.findall(r"\d*\.\d+\.\d+", t_number)
        ThreadNumber.append(t_number)
    elif "threadlength" in p_single_value.lower():
        t_len = p_single_value.split(",")[2]
        t_len = re.findall(r"\d*\.\d+\.\d+", t_len)
        ThreadLength.append(t_len)

stairs_data['riser_number'] = RiserNumber
stairs_data['riser_height'] = RiserHeight
stairs_data['thread_number'] = ThreadNumber
stairs_data['required_headroom'] = RequiredHeadroom
stairs_data['thead_length'] = ThreadLength

elif "IFCSLAB" in value:
    floor_representation = []
    f_representation = value.split(",")[6]
    floor_representation.append(f_representation)

body_representation = []
if floor_representation:

```

```

    for key in floor_representation:
        b_representation = mapped_data[key].split(",")[2]
        b_representation = b_representation.replace("(", "").replace(")", "").replace(";", "").replace("\n",
""))
        body_representation.append(b_representation)

    floor_shape_representation = []
    if body_representation:
        for key in body_representation:
            fs_representation = mapped_data[key]
            if "sweptsolid" in fs_representation.lower():
                fs_representation = fs_representation.split(",")[3]
                if ")" in fs_representation:
                    fs_representation = fs_representation.replace("(", "").replace(")", "").replace(";", "
").replace("\n", "")
                    floor_shape_representation.append(fs_representation)

    sweptArea, floorThickness = [], []
    if floor_shape_representation:
        for key in floor_shape_representation:
            s_area = mapped_data[key].split(",")[0]
            s_area = re.findall(r'#\d+', s_area)[0]
            f_thickness = mapped_data[key].split(",")[3].replace(")", "").replace(";", "").replace("\n", "")
            f_thickness = re.findall(r"\d*\.\d+|\d+", f_thickness)[0]
            sweptArea.append(s_area)
            floorThickness.append(f_thickness)

    floorLength, floorWidth = [], []
    if sweptArea:
        for key in sweptArea:
            if "ifrectangleprofiledef" in mapped_data[key].lower():

```


⁸APPENDIX H – SOURCE CODE- EXTRACTION & COSTING COMPONENTS

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Nov  7 18:26:34 2019

@author: Temitope Akanbi
"""

from pdfminer.pdfinterp import PDFResourceManager, PDFPageInterpreter
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
from pdfminer.pdfpage import PDFPage
from io import StringIO
import nltk
import re
import copy

rsrcmgr = PDFResourceManager()
retstr = StringIO()
codec = 'utf-8'
laparams = LAParams()
device = TextConverter(rsrcmgr, retstr, codec=codec, laparams=laparams)
# PLEASE CHANGE THE FILENAME HERE:
fp = open("/home/apoorv/20151218 Fields at Southpointe - Specifications - Architectural.pdf", 'rb')
interpreter = PDFPageInterpreter(rsrcmgr, device)
password = ""
maxpages = 0
caching = True
pagenos=set()
b = ""
i = 0
text = []

a = PDFPage.get_pages(fp, pagenos, maxpages=maxpages, password=password,caching=caching,
check_extractable=True)
```

⁸ Source code for the extraction and costing components

```
for page_no, page in enumerate(PDFPage.get_pages(fp, pagenos, maxpages=maxpages,
password=password, caching=caching, check_extractable=True)):
```

```
    if page_no == i:
        interpreter.process_page(page)
        b = retstr.getvalue()
        text.append(b)
        b = ""
        retstr.truncate(0)
        retstr.seek(0)
    i += 1
```

```
fp.close()
```

```
device.close()
```

```
retstr.close()
```

```
map_dict = {
```

```
    "SHEATHING" : ["061600"],
    # "GYPSUM BOARD" : ["092900"],
    # "WEATHER BARRIERS" : ["072500"],
    # "FRAMING" : ["061010"],
    # "WOOD STUDS" : ["061000"],
    # "INSULATION" : ["072100"],
    # "WALL INSULATION" : ["072100"],
    # "EXTERIOR WALL INSULATION" : ["072500"],
    # "COVERING" : ["073100"],
    # "PAINTING" : ["099123"],
    # "WALL PAPER" : ["097200"],
    # "WALL FINISH" : ["074646"],
    # "CARPET" : ["096813", "096816"],
    # "TILE" : ["093013"],
    # "WOOD" : ["096400"],
    # "STAIR" : ["064300"],
}
```

```
item_dict = {
```

```
    "SHEATHING" : ["SHEATHING", "SUBFLOORING"],
    # "GYPSUM BOARD" : ["GYPSUM BOARD"],
```

```

# "FRAMING" : ["ROOF", "FLOOR", "STAIR"],
# "WOOD STUDS" : ["WOOD STUDS"],
# "INSULATION" : ["ROOF", "BOARD", "BLANKET"],
# "WALL INSULATION" : ["INSULATION"],
# "COVERING" : ["ROOF"],
# "PAINTING" : ["PAINTING"],
# "WALL PAPER" : ["WALLPAPER"],
# "WALL FINISH" : ["WALL FINISH"],
# "CARPET" : ["CARPET"],
# "TILE" : ["TILE"],
# "WOOD" : ["WOOD"],
# "STAIR" : ["STAIR", "LANDING", "BLAUSTRAD"],
# "WEATHER BARRIERS" : ["BARRIER", "TAPE", "FLASHING", "INSULATION"],
}

final = {}

for enter in map_dict.keys():
    try:
        req = ""
        for e in range(len(text)):
            for f in map_dict[enter.upper()]:
                if text[e][-14:-8] == f or text[e][-16:-8] == f:
                    req += text[e]

        if "Section Includes:" in req:
            ind = req.index("Section Includes:")
            ind_1 = ind+17
            exm = req[ind-21:ind-18]
            lind = req.index("1.2")
            lind_1 = req.index("B.", ind_1)
            if lind <= lind_1:
                data = req[ind_1: lind]
            else:

```

```

        data = req[ind_1:lind_1]

        data= data.strip()

        data_str = ""

        for e in data:

            if (ord(e) >= 65 and ord(e) <= 90) or (ord(e) >= 97 and ord(e) <= 122) or e == "\n" or e == " "
or e == "-":

                data_str += e

        data_str = data_str.strip()

        data_list = data_str.splitlines()

        for e in range(len(data_list)):

            data_list[e] = data_list[e].strip()


j = 0
data_list_copy = copy.deepcopy(data_list)
for i in range(len(data_list_copy)):
    if len(data_list_copy[i]) <= 4:
        n = data_list.pop(i)
        j -= 1
    j += 1

if enter == "SHEATHING":
    data_list[-1] = "Sheathing joint-and-penetration treatment"
elif enter == "WEATHER BARRIERS":
    data_list[0] = "Water-resistive barrier"
n = []
for i in data_list:
    if i.upper() in req:
        n.append(req.index(i.upper()))
    else:
        continue

req_data = []
for e in range(len(n)):
    if e == len(n) - 1:
        x = re.findall("[0-9]\.[0-9]", str(float(req[n[e]-6:n[e]-3])+0.1))[0]

```

```

        if (req.find(x) == -1):
            req_data.append(req[n[e].req.index("PART", n[e]))
        else:
            req_data.append(req[n[e].req.index(re.findall("[0-9]+\.[0-9]", str(float(req[n[e]-6:n[e]-
3]))+0.1))[0]))
        else:
            req_data.append(req[n[e].n[e+1]])

req_data_1 = ""
for e in range(len(req_data)):
    req_data[e] = req_data[e].upper()
    for f in range(len(req_data[e])):
        if req_data[e][f] == " ":
            if req_data[e][f-1] == " ":
                continue
            else:
                req_data_1 += req_data[e][f]
        else:
            req_data_1 += req_data[e][f]
    req_data[e] = req_data_1
    req_data_1 = ""

rex_1 = []
for e in item_dict[enter]:
    x = re.compile(f"[a-zA-Z\-\s]+\s+(\s)")
    for f in req_data:
        rex_1.extend(re.findall(x, f))
rex_3 = re.findall("[\./0-9\s*]+\s+(?:INCH|FEET)" or "[\./0-9\s*]+\s+(?:GRADE;)", req_data[0],
re.I)
rex_4 = []
for e in req_data:
    rex_4.extend(re.findall("TYPE\s([IVX])+", e, re.I))
rex_5 = []
if enter == "INSULATION":
    for e in req_data:

```

```

#         import pdb; pdb.set_trace()

        rex_5.extend(re.findall("[a-zA-Z]+\s*FACED", e, re.I))
rex_1 = list(set(rex_1))
for e in rex_1:
    print(enter, "-->", e.strip())
for e in rex_3:
    print(enter, "-->", e.strip())
for e in rex_4:
    print(enter, "-->", e.strip())
for e in rex_5:
    print(enter, "-->", e.strip())
final[enter] = [rex_1, rex_3, rex_4, rex_5]
except:
    print(enter, "-->", "Data Not Found")

```