

BIM-based Automated Design Checking for Building Permit in the Light-
Frame Building Industry

By

Harish Narayanaswamy

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Civil (Cross-disciplinary)

Department of Civil and Environmental Engineering

University of Alberta

© Harish Narayanaswamy, 2019

ABSTRACT

Automation of the code compliance checking process has been explored extensively, particularly in recent years with the emergence of building information modelling. Still, automated code compliance checking has not yet been fully realized, as there is no standardized method for rule interpretation and building model preparation for code compliance. Manual checking of design code compliance, meanwhile, requires significant effort and time and is error-prone, while uncertainty and inconsistency in assessment lead to delays in construction process. Hence, the development of a BIM tool (i.e., an add-on software application to Autodesk Revit) to automate municipal zoning bylaw and wood framing design compliance checking for residential buildings is presented. This research also discusses the pros and cons of existing methods of code compliance checking and proposes a new classification of building code regulations for better implementation of the building rules in stages. The proposed classification is based on the complexity involved in rule interpretation and the level of difficulty involved in data extraction from the BIM model. The developed tools provide a novel, simplified framework for rules representation and for interpreting them using .NET coding language. By creating model views in Autodesk Revit of building objects based on the required elements' threshold parameter values, the add-on software application offers automated code compliance checking functionality to validate zoning bylaws related to lot dimensions based on municipal bylaws and to validate wood framing designs based on building code requirements and construction engineering specifications. A case study is presented to demonstrate the implementation of the application and its benefits compared to existing design checking approaches.

PREFACE

This thesis is the original work by Harish Narayanaswamy who completes the thesis work under the supervision of Dr. Mohamed Al-Hussein. The research related topics, proposed methodology and paper writing were finished by Harish Narayanaswamy with guidance from Dr. Al-Hussein. One conference paper related to this thesis have been submitted for publishing and it is listed as below.

List of proceedings:

Narayanaswamy, H., Liu, H., and Al-Hussein, M. “BIM-based Automated Design Checking for Homebuilders in the Light-Frame Building Industry.” Under Review (Jan., 2019) for publication in Proceedings, 36th International Symposium on Automation and Robotics in Construction, Banff, AB, Canada, May. 21-24.

ACKNOWLEDGMENTS

I would like to take this opportunity to express my sincere appreciation to the many wonderful people who have supported me during my studies. First, I would like to express my deepest gratitude to my supervisor, Dr. Mohamed Al-Hussein, for his support, direction, encouragement and support throughout this research.

I would like to express my thanks to Dr. Hexu Liu, for his continuous support through this research. Mahmud, Lana, Marko, Beda and to the all administrative and research staff of Dr. Al-Hussein's group at the University of Alberta for their care, attention, and technical support. I would also like to thank my friends Ankit, Anil, Anish, Sushmitha, Vishal, Saraswathi, Sathish and Ananthan for their help and moral support.

Finally, my deepest gratitude to my family for all the love and encouragement. Especially, I thank my Dad and brothers for always inspiring me and supporting me through everything. Also, I would like to dedicate this work to my mother, Lakshmi, for her endless love.

TABLE OF CONTENTS

ABSTRACT	ii
PREFACE	iii
ACKNOWLEDGMENTS.....	iv
TABLE OF CONTENTS	v
LIST OF FIGURES	viii
LIST OF TABLES.....	x
LIST OF ABBREVIATIONS.....	xi
FOL logical symbols meaning (Philosophy Index, 2018)	xii
CHAPTER 1. INTRODUCTION.....	1
1.1 Background	1
1.2 Problem Statement and Research Objective	4
1.3 Thesis Organization.....	6
CHAPTER 2. LITERATURE REVIEW	8
2.1 Existing Techniques in Code Checking	8
2.1.1 Corenet (Singapore).....	9
2.1.2 Statsbygg (Norway)	10
2.1.3 Design Check (Australia)	11
2.1.4 International Code Council (ICC) and General Services Administration (GSA)	

Design Rule Checking (The United States)	12
2.1.5 Other Applications.....	13
2.2 Representation of building codes	14
2.3 Interpretation of building code	16
2.4 Building model views.....	18
2.5 Compliance checking algorithms and reporting	20
CHAPTER 3. RESEARCH METHODOLOGY	23
3.1 Overview	23
3.1.1 Rule Translation.....	24
3.1.2 BIM model preparation	35
3.1.3 Rule Checking.....	36
3.1.4 Checking Report	37
3.2 Prototype Development.....	38
3.2.1 Edmonton zoning bylaw checking.....	41
3.2.2 Framing Checking.....	52
3.2.3 Domain knowledge from the standard.....	55
3.3 Object-oriented representation.....	61
CHAPTER 4. CASE STUDY	63
4.1 Bylaw Checking	67
4.2 Framing Checking	72

4.3	Discussion	78
CHAPTER 5. CONCLUSION AND FUTURE RESEARCH		81
5.1	Summary	81
5.2	Research Contributions	82
5.3	Limitations and Recommendation for Future Work	83
5.3.1	Research Limitations	83
5.3.2	Future Research and Improvements	84
REFERENCES		85
APPENDIX A: Glossary		92
APPENDIX B: User Manual		93
APPENDIX C: All the Regulations Related to Framing from Building Code		99
APPENDIX D: Extended Data Structure Developed for Prototype Excerpt from the C#.Net		110

LIST OF FIGURES

Figure 1. 1. General Structure for Rule Checking Process (C. Eastman et al., 2009a).....	3
Figure 1. 2. Building Permit Approval Process.	5
Figure 3. 1. Overview of Proposed Methodology.	24
Figure 3. 2. System Architecture.	40
Figure 3. 3. Illustration of Setback Distances Requirements for RE1 Zone (Adopted from City of Edmonton Website).....	43
Figure 3. 4. Flow Chart for checking Lot Dimensions.	44
Figure 3. 5. Flow Chart for Checking Lot Dimensions of Different Types of Houses.	46
Figure 3. 6. Flow Chart for Checking House Area.....	48
Figure 3. 7. Flow Chart for Checking of Building Coverage for Different Types of Houses.	49
Figure 3. 8. Height Consideration for Types of Houses (Adopted from City of Edmonton). ..	50
Figure 3. 9. Flowchart for checking Height of Building.	51
Figure 3. 10. Flow Chart for Framing Checking for Residential Building.	53
Figure 3. 11. Flow Chart for Checking Spacing and Maximum Height of Framing.	55
Figure 3. 12. Excerpt of framing information for checking using UML.	62
Figure 4. 1. 3D Model of Single detached house with Attached Garage (case study model 1).	64
Figure 4. 2. 3D Model of semi-detached house with Attached Garage (case study model 2).	64
Figure 4. 3. Floor Plan of single detached House with Attached Garage (case study model 1).	66
Figure 4. 4. Top view of semi-detached House with Attached Garage (case study model 2).	66
Figure 4. 5. Main User Interface of DCheck add-on.	69

Figure 4. 6. User Interface for Entering Project Details.....	70
Figure 4. 7. Bylaw Checking Report for single detached house (case study model 1)	71
Figure 4. 8. Bylaw Checking Report for semi-detached house (case study model 2).....	72
Figure 4. 9. Framing of Walls and Floors of House.	73
Figure 4. 10. User Interface for Entering Framing Details.	75
Figure 4. 11. Framing Checking Report for single detached house (case study model 1).....	75
Figure 4. 12. Framing Checking Report for semi-detached house (case study model 2).	76
Figure 4. 13. Highlighting Failed Rules Related Objects in Basement Floor.....	77
Figure 4. 14. Final Check Results After correcting all the Errors.	78

LIST OF TABLES

Table 1. Summary of Typical Literature on BIM-based Design Checking	22
Table 2. Maximum Site Coverage with Respect to Specific Housing Type and Area.	47
Table 3. Spacing and Maximum Height of Studs.	54
Table 4. Examples of Rule-based Knowledge for Checking Residential Building.....	57
Table 5. Framing Regulations from Alberta Building Code 2014.	99
Table 6. Maximum spans for floor joist table.....	109

LIST OF ABBREVIATIONS

AEC	Architectural, Engineering, and Construction
API	Application Programming Interface
BIM	Building Information Modelling
CAD	Computer-aided Design
COBie	Construction Operations Building Information Exchange
CORENET	Construction and Real Estate Network
EDM	Express Data Management
FOL	First-Order Logic
GUI	Graphical User Interface
ICC	International Code Council
IFC	Industry Foundation Class
LoD	Level of Details
RASE	Requirement (R), Applicability's (A), Selection (S) and Exceptions (E)
SMC	Solibri Model Checker
SQL	Structured Query Language
SWRL	Semantic Web Rule Language
XML	Extensible Mark-up Language

FOL logical symbols meaning (Philosophy Index, 2018)

\neg	negation (NOT)	The tilde (\sim) is also often used.
\wedge	conjunction (AND)	The ampersand ($\&$) or dot (\cdot) are also often used.
\vee	disjunction (OR)	This is the inclusive disjunction, equivalent to and/or in English.
\oplus	exclusive disjunction (XOR)	\oplus means that only one of the connected propositions is true, equivalent to either...or. Sometimes $\underline{\vee}$ is used.
\downarrow	alternative denial (NAND)	Means “not both”. Sometimes written as \uparrow
\downarrow	joint denial (NOR)	Means “neither/nor”.
\rightarrow	conditional (if/then)	Many logicians use the symbol \supset instead. This is also known as material implication.
\leftrightarrow	biconditional (iff)	Means “if and only if” \equiv is sometimes used, but this site reserves that symbol for equivalence.
\forall	universal quantifier	Means “for all”, so $\forall xPx$ means that Px is true for every x .
\exists	existential quantifier	Means “there exists”, so $\exists xPx$ means that Px is true for <i>at least one</i> x .
$()$	parentheses	Used to group expressions to show precedence of operations. Square brackets [] are sometimes used to clarify groupings.

CHAPTER 1. INTRODUCTION

1.1 Background

Technological advancements in the Architectural, Engineering, and Construction (AEC) industry have digitalized nearly every stage of the building lifecycle, and this digitization has been a significant advancement in the industry over the past several decades. Automated code compliance checking saw a major leap with the advent of Building information modelling (BIM) in the late nineties (Han, Kunz, & Law, 1998). BIM has been the major technological advancement in the AEC industry that has most benefited AEC professionals over the traditional computer-aided design (CAD) approach in every stage of the work, starting from design, execution, and management of construction activities with a digital form of information in three-dimensional geometry and semantics of individual elements in the form of objects. (C. Eastman, Lee, Jeong, & Lee, 2009a). In many countries, such as Singapore, England, and France, BIM has been made mandatory for government projects.

Through many years of research, many researchers believe that the full benefit of BIM technology is obtained when it is applied from design to demolition stage, while the use of BIM in every construction process will improve the overall efficiency. With the increase of complexities in the construction process, getting the correct information for the right task will help in getting better results. With the development of the neutral format know as industrial foundation class (IFC), project development can fulfil the condition of fragmented planning tasks, and this led the automatic parametric generation of design and automation checking of the design (Ismail, Ali, & Iahad, 2017).

BIM technology has been adopted in the AEC industry for the designing of building models, which can be utilized for a wide variety of applications across the building lifecycle. Automated code compliance checking system is one of the application processes for checking the models

in accordance with building codes, regulations, and bylaws with the use of BIM data models. Automated rule checking came to light well before the introduction of BIM technology with use of 2D CAD drawings. Even while so many authorities and researchers have been working on the automated code compliance checking process for many years, it is not yet completely in use it is still a semi-automated process. Many researchers have developed an application for safety, egress, and design checking, but still no application is in use efficiently, even in some countries where the BIM model for the design checking process was made mandatory. Singapore, the UK, and the United States use BIM model for automated checking with online submission of IFC file for approval purposes. But there have been no updates in many applications since first developed, and even a fewer number of those have survived.

Most of the building code and bylaw compliance work has been done manually by the professionals in the construction industry, which requires more manpower, and is an error-prone, time-consuming task that will not be consistent. Automation of checking, where well-defined rules can be applied automatically with minimum user involvement, is increasingly needed. With the complete translation of the human-readable natural language code into computer interpretable language, with suitable methodologies for implementation and clear understanding of building regulations by the logical representation of regulations, will result in a good process for the compliance. As many researchers have been involved in this area of research for many years, the main methodology for implementing this process of transferring building code into machine-readable format involves four main steps as follows: rule interpretation, building model preparation, rule execution, and rule reporting, as shown in Fig 1. Each step carries utmost importance as it holds different responsibilities and functions.

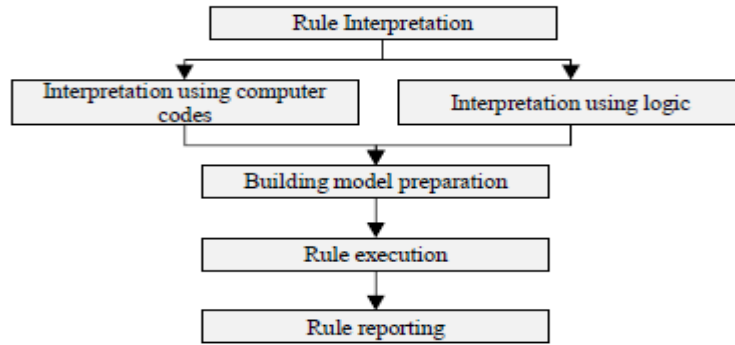


Figure 1. 1. General Structure for Rule Checking Process (C. Eastman et al., 2009a).

Rule interpretation is the first step, in which the originally produced human language will be interpreted into a computer-readable format. Classifying the building code into types based on the level of difficulty in translating the natural building code into computer interpretable format and the level of difficulty in extracting the information from the BIM model will facilitate the interpretation of the code in stages based on classification for complete automation. This step is the most critically important step in developing an automated code compliance checking application. There are different ways of translating the building code and extract the required information from BIM model. Next, the model is created using BIM technology with the required level of details and the required model views for extracting model information. The next step is rule execution where encoded rule and model are brought together for code checking, and finally, results are generated in the reporting stage based on the compliance with rule outputs in text-based reports.

Understanding what amount of information is required for an element in the model plays a very important role in ensuring its utilization in code compliance checking. Level of details is how many details are included in model objects related to dimensional, special, quantitative, qualitative, and other data included in the model element to support required purposes. The American Institute of Architects (AIA) has published a framework for level of development required for model element content. There is different level of development known as level of

details (LOD): LOD 100, 200, 300, 400 and 500. The LOD for each level are defined as follows:

LOD 100: The building elements are developed to represent the information on a basic level, graphical representation of building models is done in this stage.

LOD 200: The building elements are developed with approximate quantities, size, shape, location, and orientation. Non-graphical information can be attached for model elements.

LOD 300: The building elements are developed with accurate modelling and shop drawings, where elements are defined with specific assemblies, precise quantity, size, shape, location, and orientation. Non-graphical information can also be attached to model elements.

LOD 400: The building elements are developed with specific assemblies, with complete fabrication, assembly, and detailed information, in addition to precise quantity, size, shape, location, and orientation. Non-graphical information can also be attached to model elements.

LOD 500: The building elements are modelled as constructed assemblies for maintenance and operations, in addition to field-verified details in terms of size, shape, location, quantity, and orientation. Non-graphical information may also be attached to model elements.

1.2 Problem Statement and Research Objective

Every building to be built must go through a process of assessing whether it meets legal requirements based on the building code, regulations and bylaw compliance. The process of checking the building design model is still manual. The manual application process to obtain the approval and permit to proceed for construction work is shown in Figure 2. And when it comes to the framing for residential buildings, the modelling of the framing will be done with reference to the wood framing construction documentation, except in some special or critical conditions. Validation of framing is done manually, which is time consuming, error prone and not consistent.

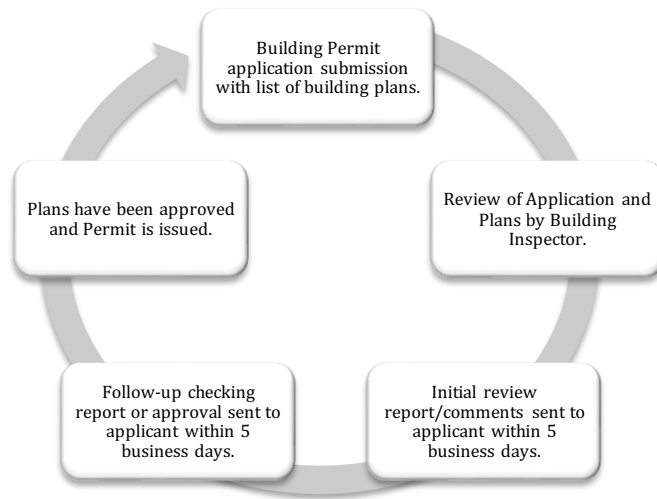


Figure 1. 2. Building Permit Approval Process.

And it is the same with the approval for the building permit, where 2D drawings have been used for the zoning approval for the construction process, which adds more time to the processing of permit approval, leads to a delay in start of the project, and if any corrections have to be done in design, it will become difficult to identify those when comparing with old drawings.

The major problem in the process of automated checking is the rule interpretation, where all the human-written codes must be translated to computer-interpretable format. All the building codes are not self-contained and make reference to various other documents that all industry professionals should be familiar with. There are many ways to translate the building codes, as shown in Figure 1. With interpretation using either computer codes or logic, some simple rules are easily transferred with minimal efforts, but the difficulty level increases with rules or codes that are difficult to define in computer code or logical format.

This research is based on the following hypothesis:

“Using object-based representation and with classification of building rules, simplified BIM object model views can be generated to develop an automated checking application in an

efficient and comprehensive way.”

This research involves the development of a BIM-based automated design checking prototype in the form of a software application (an add-on for Autodesk Revit) for Edmonton zoning bylaws, used to verify compliance of lot design for residential house construction depending on the residential zone, and to check that the wood framing design for walls of a residential building are in accordance with Alberta Building Code 2014 Part 9. The automated checking software application relies on the representation of the building rules based on building objects and depends on the classification of the building rules, where the classification is based on how convenient the code is to translate into computer-readable format, the level of difficulty involved in the extraction of information from BIM models, and then the representation of those building rules in logical form, which helps in translating to computable format effectively and more accurately. The first type of classification of rules includes data that can be easily accessed from BIM model. The second type of classification requires some expert knowledge where the required information should be derived from BIM model. The third type of classification of rules are the ones that need to be simplified and analyzed, and the information requires an extended data structure.

The DCheck add-on software application developed for Autodesk Revit to accomplish compliance checking for zoning and framing in residential building design can be used at any point during the design process for checking the model so that the designer can correct any errors if present during the design process.

1.3 Thesis Organization

This thesis consists of five chapters. Chapter 1 (Introduction) introduces the topic and research objective and provides an overview of this thesis. Chapter 2 (Literature Review) provides a review of the literature gathered about the topic, and background information covering the development of technology in automated code checking process up until now. It also provides

insight into rule classification based on the rule interpretation for automation and different design checking applications developed by authorities and researchers. Chapter 3 (Research Methodology) presents the methodology used in this research, which consists of four main elements: (1) rules translation, (2) BIM model preparation, (3) rule checking, and (4) checking report. Chapter 4 (Case Study) is a validation of software application using a case study presented here and discussions about adoption of automated compliance checking. And Chapter 5 (Conclusion and Future Research) is where conclusions and future scope of research are presented.

CHAPTER 2. LITERATURE REVIEW

This chapter presents the critical review with respect to code checking techniques, building code representation, building model view, and compliance checking algorithms in order to clarify the point of departure for this research.

2.1 Existing Techniques in Code Checking

Exhaustive studies have been made in automating the building code by many researchers in the field, each exploring different techniques in interpreting the rules from their perspectives. Fenves initiated research that examined how to logically organize the rules and regulations in the 1960s, where he structured the regulation data in a decision table. His efforts in classifying the rules into decision tables, combined with those of other researches who followed him, made progress in this area and has led to this stage in code compliance (Johannes Dimyadi & Amor, 2013). With the use of CAD tools for design purposes by AEC professionals in the 1990s, the automation of design checking has gained more interest among researchers, and their research has included the development of the following: logic-based approaches for the organization of design standards (Rasdorf & Lakmazaheri, 1990); computer representation of design standards (Fenves, Garrett, & Kiliccote, 1995); and knowledge-based expert systems capable of reviewing building design (Dym, Henchey, Delis, & Gonick, 1988). With the evolution of BIM technology and the use of standardized IFC file format through the AEC industry, the automation of code compliance was made more convenient. So many organizations around the world have been involved in making this process automated. In some countries, the BIM model is compulsory for the purpose of building approval, and some countries are trying to make it mandatory in upcoming years.

Following are the list of applications developed by different countries and government authorities for automated compliance of their country's building code.

2.1.1 Corenet (Singapore)

In 1995, the building construction authority (BCA) of Singapore initiated CORENET (Construction and Real Estate Network) as a comprehensive network system with a series of IT systems for exchange of information between government agencies and parties involved in construction and real estate (Global Reporting Initiative, 2014). CORENET for approval process provides electronic web-based submission system incorporating in-house building plans (BP) expert system to check 2D plans for any technical irregularities with reference to the building regulations (Preidel & Borrmann, 2015). E-PlanCheck, as part of CORENET, was the first initiative developed for automated code-checking (Malsane, Matthews, Lockley, Love, & Greenwood, 2015). In 2002, BCA updated the system to CORENET e-PlanCheck with 3D model. CORENET consists of three platforms: e-submission, e-PlanCheck, and e-info (S. Zhang, Teizer, Lee, Eastman, & Venugopal, 2013). Project-related plans and documentation will be submitted to regulatory authorities through e-submission for approval of building plan, structural plan, temporary occupational, safety certificate and so on. E-PlanCheck automatically checks the electronically submitted models for compliance of regulations using BIM, and information regarding the codes, regulations, guidelines, standards are provided in e-info (Khemlani, 2005).

Higher level of semantics that are relevant to code checking requirements are added to basic building model information from IFC through FORNAX and independent platform used by e-PlanCheck (Khemlani, 2005). FORNAX is an object library made by encapsulating building components into objects, where an object contains relevant attributes code and rules apply to that. Development of this application was the result of earlier efforts in this field made by Singapore's government authority to translate the building code into computer-readable format for automated building code compliance checking (W. Solihin & Eastman, 2015), and was used as a pilot project in Norway and New York with replacement of rules required by Norway and

by using ICC (International Code Council) codes for New York.

2.1.2 Statsbygg (Norway)

The Norwegian government organization, Statsbygg, acts as the Norwegian government's key advisor in construction, building commissioner, property manager and property developer (“STATSBTGG website,” 2018), CORENET e-plan checking system has been used for a couple of IFC-based BIM building projects as an early effort by Norwegian authorities (“Automated compliance checking using building information models,” 2010). Multiple platforms, like e-PlanCheck, SMC, dRofus, and EDM model Checkers, were also adopted for the purpose of experimenting for finding a better checking system. “HITOS” is a BIM project managed by the Statsbygg governmental agency and Tromso University since 2005, for which several software have been used for modelling architectural, structural, MEP, cost estimation, and energy simulation, and EDM model server was also used for storing and accessing the model data in IFC format (Malsane et al., 2015). dRofus was used for spatial requirement checking: dRofus is a database system used for managing architectural, equipment’s for early stage planning and through project and technical/functional requirements. dRofus provides overview and detailed room, department and area information, room data sheets and building elements planning (“STATSBTGG website,” 2018). Solibir model checker (SMC) was used for checking accessible design of the model. SMC was developed in 2000 in Finland as a quality assurance and validation tool. Developed into a stand-alone graphical driven rule-based compliance checking and reporting application, it was built with a set of rules managed by ruleset manager and can only be customized in a limited way by changing parameters (Malsane et al., 2015). Solibri IFC-based universal design checking implementation could reduce common design failures or deficiencies by 60 to 70% (“Automated compliance checking using building information models,” 2010). The user can configure rules using a parametric table structure for the requirements of code and to validate the model. SMC has its rules that are

based on JAVA; most of the rules are hardcoded into software, and it is difficult to specify new types of rules and even to modify hardcoded rules. For HITOS project, the accessibility rules have been translated into parametric table structure, where the end-user can input the parameter values for the rules to check if the national or other code change from ISO standards. SMC, as described above, uses not only geometry of single object, it also considers the other associated objects and their properties. And the final reports are displayed in graphical or several documentation file formats, which gives the results in three levels: critical, moderate, and low.

2.1.3 Design Check (Australia)

DesignCheck was developed by Australian authorities for automated building code compliance for Australia focus on accessible design regulations (Ding, Drogemuller, Rosenman, & Marchant, 2006). Code checking efforts by Australia involves development in two phases. The first phase was to assess the capabilities of existing rule checking systems to find out which would be the best one for computerization of Australian standards (Ding et al., 2006). Both SMC and Express Data Management (EDM) were considered as possible platforms for automated code checking. EDM was considered as the more suitable one because of its ability to provide a publicly accessible definition language to represent building codes. After the first stage of checking for feasible solution, Different domain-specific knowledge can be encoded to EDM rule base and can be applied to check a building model (Mike, Automated, & Drogemuller, 2004). ArchiCAD modelling software, which supports BIM, was used for modelling purpose, and IFCTreeView approach in ArchiCAD allows the user to select the element to define extended properties required by the codes (Mike et al., 2004). An internal model has been developed, that extends the IFC model for compliance of large scope of interoperability of architectural, structural, fire engineering and building service domains and object-based rules have been used by EDM database for process of mapping required to translate CAD model to the IFC2×2 model and then to DesignCheck internal model (Ding et

al., 2006).

A graphical approach was taken for accessibility checking, where spaces are accessible to previously checked adjoining space are defined as accessible. The report of checked rules is generated in text-based format and saved into XML and HTML documentation. Checking can be made by clauses of the code or by type of object. It has the ability to check the model at various stages in the design process, as it has a rule schema for early and detailed design stages, as well as for specification. Because of this, it is more targeted to architects and designers rather than just building control certifiers (Ding et al., 2006).

2.1.4 International Code Council (ICC) and General Services Administration (GSA) Design Rule Checking (The United States)

Studies on automating code compliance by the United States authorities began around 2000. GSA, an independent agency of the United States government (“Automated compliance checking using building information models,” 2010), issued BIM-guidance in 2006, and from 2007 made it mandatory to have a BIM model for validation for all the projects seeking permission for spatial planning projects. The application uses the SMC platform and Design assessment tool for extending rules, developed by Georgia Institute of Technology. The US court design guide (CDG) has been used for the spatial rules that have been translated into parametric tables in SMC platform. Building model elements are mapped to graphical nodes, where two methods have been used for checking: (1) topological graph checks for routing path by connecting between spatial elements, and (2) the metric graph represents distances based on human movements, and is used to know the distance between two spaces (C. Eastman, Lee, Jeong, & Lee, 2009b). The most interesting initiative in this area is SMARTcode, which was started in 2006 and handled by ICC, a US-based association that develops the master building codes for residential and commercial buildings and most institutional buildings. In 2005, the ICC board approved an investment in making automated code checking for international code

in conjunction with AEC3 and Digital Alchemy, which is called SMARTcodes (“AEC3 website,” 2012). SMARTcodes is a project for transforming natural language code into computer interpretable format, and a dictionary of the properties found within the building codes have been developed, which helps to reduce the errors in interpretation by facilitating a search to determine only those that are relevant to topic and to deliver these exclusive of all the other non-relevant codes (See, 2008). The dictionary is also helpful in communication between SMARTcodes model checking system and the IFC building model (C. Eastman et al., 2009a). SMARTcode represents the code in object properties in XML form, and this provides a significant platform to carry mapping between IFC building model for checking. ICC allows the end-user to check through the website, where it requires the input of building model and details related to building location, code to be checked and model checking system. It is limited to some pre-configured building models. The final report of checking is provided in several formats, such as PDF, XML, RTF, XLS and HTML, and table-based summaries and graphical-based reports are also available for analysis report (“AEC3 website,” 2012).

2.1.5 Other Applications

Apart from the above-explained applications, many researchers have developed applications with different interpretability techniques covering different aspects of code compliance checking. LicA is an application that performs the automatic code checking for the Portuguese domestic water system regulation (Martins & Monteiro, 2013): it accomplishes the checking of water network by nodes that represent flow segments (Poças Martins & Abrantes, 2010), and the hydraulic analysis results of each node are computed and checked for the regulations. Fall hazard protection, which comes under safety checking, building models are checked for the safety issues related to fall from heights (Zhou, Whyte, & Sacks, 2012). This automated software identifies the dangerous activities in project schedules and areas in the building where hazards appear and processes protective activities to improve the existing process based on the

models designed by providing textual and graphical reports, and warns when guardrails are missing, partially removed, or incomplete. The rules for safety in construction are checked from Occupational Safety and Health Administration (OSHA) of United States Department of Labour. Safety compliance checking for fall protection design and planning is examined in a study by Melzner et al., the authors examined a customizable automated safety checking platform by developing an application as an add-on to BIM software, where the regulations from both the USA and Germany regarding fall protection have been integrated into platform. Preventive safety equipment is designed, estimated and included in the construction schedule before construction starts, and further visualization of safety information is developed (Melzner, Zhang, Teizer, & Bargstädt, 2013).

Implementation of rules and regulations for building design checking is not always straightforward because ambiguities and other imperfections found in the regulations may hinder objective data interpretation by the computer, or because manual interpretation of design information may be required; this reveals that a fully automated code compliance checking process requires improved fully machine-interpretable building code (N. O. Nawari, 2012). These kind of issues in LicA have been addressed by creating different categories for the compliance checking results and by creating a class for checks that were performed but should be reviewed manually (Martins & Monteiro, 2013).

2.2 Representation of building codes

As the complexity in building design and building construction processes is increasing with new creative designs and with the use of new technologies in construction, the need for automatic model checking is becoming more pressing with advancement in other areas of construction. The representation of code in machine-readable format should possess enough elasticity and expressiveness for efficient compliance checking (N. Nawari, 2012). All the applications of code checking discussed so far, all use independent regulatory data for

representation of building rules either directly or via other dependent systems where representation is hardcoded into the machine-readable format, which is subjected to manual updates by the software developers. As observed by researchers, the representation of some unique concepts within the code is not always well-defined (Clayton, Fudge, & Thompson, 2013). Studies made by Eastman on the classification of building codes for efficient way of translating building code, have classified the rules into four different categories (W. Solihin & Eastman, 2015). They are as follows: 1) Rules that require a single or small number of explicit data, where explicit attributes and entity references that exist inside the dataset are checked; 2) Rules that require simple derived attribute values where checks are conducted on a single value or a simple set of derived values; 3) Rules that require extended data structure, for example, when an extension to data structure that encapsulates higher level semantic condition of the building data is required in this class and involves complex requirements for code checking; 4) Rules that require a proof of solution, for example, the kind of rules that do not require the check for compliance or non-compliance, but rather require a proof of solution. Performance-based results are generally represented in this class where the focus is more on how the building model proves compliance rather than just satisfying prescribed criteria.

Applicability of the rule checking system can be categorized into different code checking categories based on the applicability of different type of code conditions (W. Solihin & Eastman, 2015). The codes that are applicable to all buildings can be general building codes by national, regional or municipality level of organizations. Codes that are best for the workflow practices within design or engineering firms are the rules that are defined by the clients' organizations, and defined by programmatic requirements for buildings made by design firms, such as space requirements, circulation issues, special site considerations and some that are defined during project design and construction (C. Eastman et al., 2009b). The scope of rules within this type fall into different categories, in general they are as follows (W. Solihin

& Eastman, 2015):

(1) Checking for well-formedness of building model, where rules are concerned primarily with syntactic aspects according to the standards or required set of conditions for model views. (2). Checking for building regulatory, where building codes are well-defined or prescriptive building regulations. (3). Checking for client requirements, where buildings are designed for a specific purpose like hospital or courthouse. (4). Checking for constructability and other contractor requirements, where rules involve temporary objects or those present only during pre-construction process. (5). Checking for safety, where there are support decisions to eliminate the potential danger to workers during construction and maintenance staff operations. (6). Check for warranty approvals. Post-construction issues related to warranty or the cost to maintain. (7). Checking for BIM data completeness for handover to the facilities management (FM). BIM data modelling for FM through the lifecycle is often not considered earlier in the design process in most of the cases, such as the information defined by COBie and other families of information exchange (IE) (Macit İlal & Günaydın, 2017).

2.3 Interpretation of building code

The crucial part in automated compliance checking is the first step, which is rule interpretation, different technologies have been applied to transferring the natural language code into machine-readable format. A visual programming language (flow-based) called “visual code checking language (VCCL)” Cornelius et al. (2015) used to overcome the complexity and insufficiencies of existing approaches. With visual language, users who are not familiar with computer programming can easily approach code compliance using visual symbols that are connected and nested to automatically generate a machine-readable building code (Kim, Lee, Shin, & Choi, 2018). The visual language is a representation of modular system of signs and rules using visual elements instead of textual ones where the users will have transparency and visibility of the processing and verification and validation can be done simultaneously or

trailing plausibility checks (Preidel & Borrmann, 2015). The known visual programming software applications for building design are grasshopper for Rhinoceros3D (“Rhinoceros3D., website,” 2018) or Dynamo for Autodesk Revit (“Autodesk Inc., website,” 2018). Conceptual graph representation of the rules is useful for eliminating ambiguities in interpretation for building permit, where it provides a template for analysis and breaks down the complex rules into easily understandable atomic rules and constraints (Wawan Solihin & Eastman, 2015).

Object-Based representation of the building code has been applied by many researchers with different scopes with respect to automated checking, examples of which are as follows: Object-based representation of code in XML language (Tan, Hammad, & Fazio, 2010) for checking the envelope design where building codes are grouped as decision-tables; DesignCheck (Ding et al., 2006), which also uses object-based representation of elements properties; and, the semantically rich object model (Malsane et al., 2015), which was developed for fire safety for dwellings and houses using England and Wales’ building regulations with pre-checking application for completeness of information, as well as compliance at any stage of project and consistency check for building regulation. High-level logical rules-based mechanisms with low sentence-centred approach according to type of object and properties for Korea building permit (Park, Lee, Lee, Shin, & Lee, 2015), where three types of method classification have been done: (1) Divides type of instance, (2) Type of property, (3) Content of checking. These methods are then combined to form an intermediate pseudo-code, which will be later parsed into computer interpretable format. KBimcode, a computer-readable script language of the Korean building code, is carried out in steps from original code sentence, atomic sentence, translated atomic sentence (TAS), configuration extraction from TAS, arithmetic logic unit (ALU) and finally, expression of methods and relation, where the conditions and threshold value to be validated have been expressed in machine-readable format by the end of these steps (Lee, Lee, Park, & Kim, 2016).

Rule-based algorithms have been implemented on top of commercially-available BIM platforms (S. Zhang et al., 2013), wherein the rules are simplified from the natural language into computable format with different colour patterns to identify the objects, object attributes, and prevention system are transferred into algorithms for checking. Context-free grammar (CFG) in natural language processing and classification of morphemes can be categorized into four types: (1) Object (noun), (2) method (verb), (3) strictness (model), and (4) others. Rules are transferred into computer interpretable format automatically by analysing the sentence based on above classification (Uhm et al., 2015). semantic natural language processing techniques and express data-based techniques (J. Zhang & El-Gohary, 2017) to extract and transfer text document automatically into semantic logic based information representation. This system uses three main modules: (1) “a regulatory information extraction and transformation module” which translates the building codes into logic rules using semantic NLP algorithms, (2) “a design information extraction and transformation module” which transforms the extracted information into logical facts using EXPRESS data processing-based algorithms, and (3) “a compliance reasoning module” which automatically reason about the compliance of logic facts with logic rules using semantic-based logic reasoning algorithms (J. Zhang & El-Gohary, 2017).

2.4 Building model views

Retrieving required information from BIM model, which is the important step in automating process. For rule execution, both rule interpretation and building model information must be accurate, and the user should provide the BIM model with all the elements properties information of building model for check because most of regulation compliance checking are based on object properties in the model that can be extracted in many ways (Choi, Choi, & Kim, 2014). The query-based extraction of information can be best suited to providing information to support decision making processes (Lawrence, Pottinger, Staub-French, &

Nepal, 2014). Querying with GML (Geographical Mark-up Language) schema (Nepal, Staub-French, Pottinger, & Webster, 2012) for location, and to check construction-specific spatial information, provides rich representation of construction-specific information compared to existing BIM tools, which then can be integrated in a common XML format for checking. ifcXML schema is more helpful with XML-based interpretation of building codes for extracting information from the models (Shih, Sher, & Giggins, 2013). Automated code conformance checking (AC3) framework developed by Nawari et al. uses the abilities of LINQ to XML as in-memory programming platform. Where LINQ provides query experience across different data model, the ability to add more data models within query and flexibility of encoding unlimited range of rules increases the interest in interoperability potential of XML (O. Nawari & Email, 2011).

Object-based building model representation (Yang & Xu, 2004) examines the issues of object-based representation of code provisions Industrial Foundation class (IFC) with classified rules based on the object-based rule representation classes with their encapsulated attributes and methods. Semantic object model developed on IFC methodology with additional entities/types and reach set of IFC properties can meet the requirements to comply with code. Element view representation of the building code will help in understanding the impact of building code clauses on individual building objects and this is an easy way to maintain the relation between building objects and their related regulations (Malsane et al., 2015).

Semantic technology approach provides building information in a widely interoperable format based on logic theory (Hjelseth & Nisbet, 2010), i.e., graphical and several rule languages that are available in semantic web domain to express logic into rules. Where direct deployment of a declarative implementation can be done, the rule languages enable for better definition of building regulations and standards with little need to write procedural code. The ontological aspect is challenging when extracting the information because it is difficult to establish a

naming convention for wide use compared with linguistic approach (J. Dimyadi, Solihin, & Hjelseth, 2016). The key aspects for successful implementation of intelligent applications through BIM and ontology technology are as follows (Chen & Luo, 2017): (1) ontologies developed should be in accordance with specific domains and ensure accuracy and completeness of information; (2) ontological representation of heterogeneous data included should be accurately described; (3) a standardized and reliable approach of establishing and implementing SWRL rules. Web Ontology Language (OWL) ontology for IFC (ifcOWL) allows the user to efficiently model and manage distributed data even with poorly modelled inter-document references in IFC, because via OWL one can use general-purpose reasoning tools without developing specific system for each data model (Ebrahimipour & Yacout, 2015). The representation of well-established IFC data for construction data, where the notion of data type is different from EXPRESS and OWL data types, the standard ifcOWL ontology from the EXPRESS schema of IFC for usable and recommendable ifcOWL ontology through the industry it should remain in OWL2 DL, and should match the original EXPRESS schema and should be used primarily to allow IFC file conversions into RDF graphs. The OWL class expressions are used to improve robustness of ifcOWL representation for better integrity, consistency and applicability (Terkaj & Šojić, 2015).

2.5 Compliance checking algorithms and reporting

Extensive studies have been made in the field of automated code compliance checking by many researchers around the world, and it started with the logical organization of rules and regulations by Fenves et al. (1987), followed by the structuring of regulations in decision tables. Later, with the emergence of building information modelling, complete automation of building code checking seems achievable. Various technologies in automating code compliance checking process using BIM technology have been summarised in below Table 1. The rule interpretation process is the most critical stage in the field of automated code compliance,

where various technologies have been investigated and employed. Even with so many technologies available, there is no standardized method for translating the complete building rules and regulations into computer-readable format. BIM model preparation being the second step, where the building models will be developed with BIM technology-enabled software tools with certain level of details. The code compliance checking process can be made more efficient by defining the required level of details (LOD) to which building models should be developed. With many technologies in use for automating the checking process, there should be a standardized technology for the efficient and comprehensive translation of rules and regulations, and checking applications should be designed in such a way that updates or changes, in accordance with updates to building codes and bylaws, are easy to accomplish. Until now, even after the development of so many applications for automated code compliance checking, there is no single application that is consistent and efficient in completing the compliance checking process.

Table 1. Summary of Typical Literature on BIM-based Design Checking

Article	Checking Platform	Focus	Research Theme		
			Code representation	MVD	Checking algorithms
Khemlani 2005	FORNAX	Rules in Building plans and services (Singapore)	Computer code	IFC based (FORNAX)	Object-based approach
Preidel and Borrmann 2015	CodeBuilder plugin	German fire code	VCCL (visual code checking language)	VCCL graph	Flow-based visual Language
Pauwels et al. 2011	Semantic web	Acoustic Performance Checking Occupational circulation rules	Semantic rule language	N3Logic rules	Semantic Web Ontology language
Martins & Monteiro, 2013	LicA	Portuguese Domestic Water system	XML-based parametric tables	IFC based	Structured Query Language
Sjøgren 2007	SMC (Solibri model checker)	Norway's Building accessibility rules	Parametric tables	IFC based with adding geometric data	Object-based parameters checking
Zhang et al. 2013	Rule checking Process	OSHA Fall protection and safety	Parametric tables	Object-based parametric model	Object-based and logic approach
See 2008	DA's SMART codes for SMC, AEC3 XABIO	United States: ICC Building code	SMART builder	IFC based	RASE
Ding et al. 2006	EDM	Australian Design checking for disabled access code.	Rule-based language	IFC based using internal model schema.	Object-based approach using ExpressX language
Tan et al. 2010	Rule Engine	Building Envelope (Canada)	XML based decision tables	EBIM, XML based model	Decision table
Lee et al. 2015	KBIMLogic program	Korean Building code	Parametric table	Object based parametric model	Logic-based query

CHAPTER 3. RESEARCH METHODOLOGY

This research is to automate the building permit approval based on BIM-based design checking. This chapter illustrates the proposed method in detail and offers a clear explanation of the development of the prototype software application.

3.1 Overview

Figure 3.1 shows the overview of the process of automated design checking of building code and municipal bylaws for residential buildings. The outlook of this process involves the input of building design data from BIM models, which are designed in accordance with building codes and municipal bylaws and involves the mechanism of developing code checking functions and BIM model preparation in Revit software and the output of the process is the final report regarding the compliance checking. This process is then divided into four main steps: (1) Rule translation, which is the interpreting of natural language building rules into computer interpretable format; (2) BIM model preparation, which involves the designing of the building model in Autodesk Revit software and creating model views for extracting the information from that model; (3) Rule checking, which involves the checking of the designed model with the encoded rules; and (4) Checking report, which is where the compliance check result is obtained.

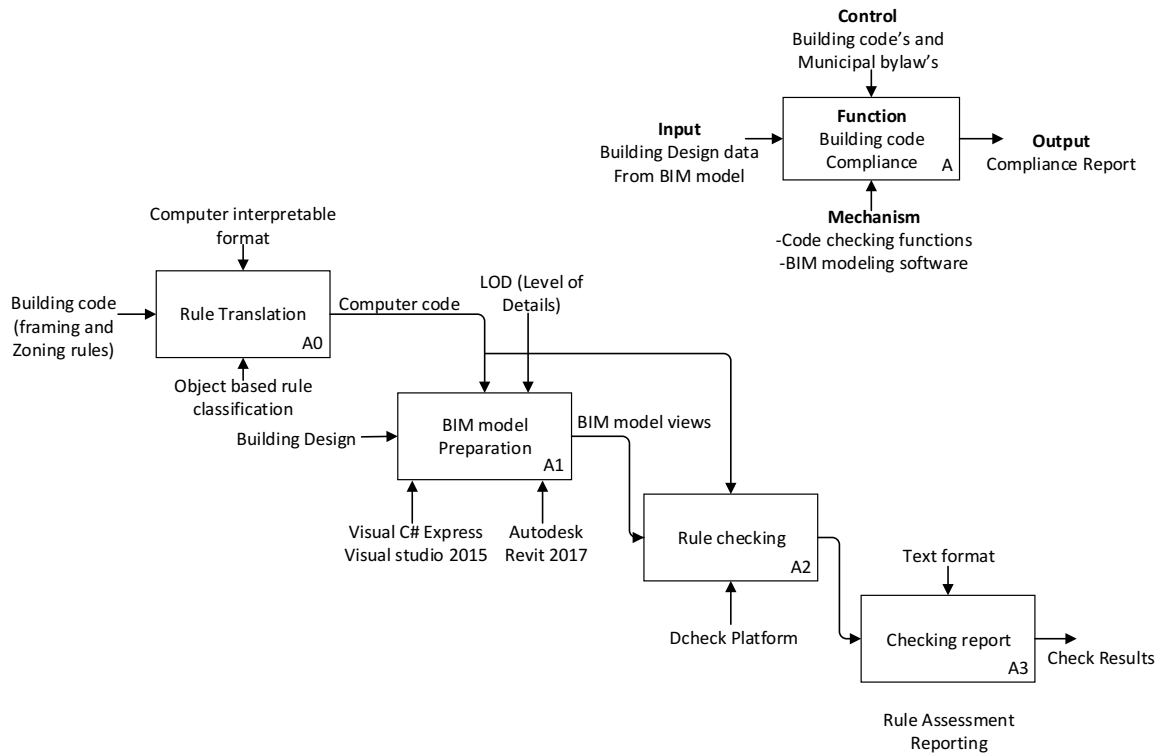


Figure 3. 1. Overview of Proposed Methodology.

3.1.1 Rule Translation

Computer-readable representations of the context and content of the building code related to wall framing and bylaws related to zoning conditions for residential buildings are developed given that the variables δ_{ss} , δ_{lot} , δ_{BC} , δ_{TH} and δ_{SB} are stud spacing, lot dimension, building cover area, total height of building, and minimum setback distances, respectively. The building regulation conditions δ_{ss} , δ_{lot} , δ_{BC} , δ_{TH} and δ_{SB} are represented in logical form as shown in the equations (1), (2), (3), (4), and (5) below for translating the building regulations conditions into C# programming language. If the equations return a value of 1 then the respective regulation has failed to satisfy the condition, else regulation will be in accordance with building codes and bylaws. Each regulation has a different level of complexity depending on the variables i , j and k as shown in equations (1, 2, 3, 4, and 5) for translating them into computer-readable format and depending on the accessibility of the required information from the BIM model for satisfying the conditions δ_{ss} , δ_{lot} , δ_{BC} , δ_{TH} and δ_{SB} . Building rules must be translated into computer-readable format with semantically rich and object-oriented information that

understands the domain knowledge of building characteristics for comprehensive automated checking process. In this study, object-oriented programming language (i.e., C#) is used for compliance checking because of the flexibility and consistency this language offers with respect to encoding building rules related to wall framing (SS_{bc} – maximum studs spacing) and municipal bylaws related to different zones (L_{bylaw} – minimum lot dimensions, BC_{bylaw} – building coverage area, TH_{bylaw} – maximum building height, and SB_{bylaw} – set back distances) and accessing BIM model information for checking (L_{BIM} - lot dimensions, BC_{BIM} - building coverage area, TH_{BIM} - building height, SB_{BIM} - set back distances and SS_{BIM} - studs spacing). Rules are classified into three types: easy, intermediate, and difficult based on the complexity in variables i , j and k to interpret and retrieving information from BIM model (L_{BIM} , BC_{BIM} , TH_{BIM} , SB_{BIM} and SS_{BIM}). Building rules (L_{bylaw} , BC_{bylaw} , TH_{bylaw} , SB_{bylaw} and SS_{bc}) are represented based on the building objects, so that it will be convenient to know which information needs to be extracted from the building model required for compliance. As shown in Figure 3.1, Rule Translation is the first step in the automation process, where regulations associated with framing SS_{bc} from the 2014 Alberta Building Code, and zoning regulations L_{bylaw} , BC_{bylaw} , TH_{bylaw} , and SB_{bylaw} from Edmonton Municipal Zoning bylaws are taken as input in human-readable natural language format. These regulations L_{bylaw} , BC_{bylaw} , TH_{bylaw} , SB_{bylaw} and SS_{bc} from the building code and bylaws are then represented based on building objects as shown in Table 5, where the regulations have been represented in detail based on building objects attributes (LA^{bylaw} – lot area, LD^{bylaw} – lot depth, LW^{bylaw} – lot width, PB^{bylaw} – building area, AB^{bylaw} – accessory building area, FSB^{bylaw} – front set back, SSB^{bylaw} – side set back, and so on) with conditions and threshold values for L_{bylaw} , BC_{bylaw} , TH_{bylaw} , SB_{bylaw} and SS_{bc} to be satisfied with operator. These rules are translated into computable functions like `CheckLaneAbutting`, `CheckSetBack`, `CheckWallsstudSpacing` and so on, with all the conditions to be satisfied being encoded into these functions. Threshold values of all rules are defined

separately, so that if required, can be easily changed to accomplish automated checking for different jurisdictional bylaws. Even users with basic coding background can easily access this and make changes if required. Below is an example of some predefined threshold values related to Edmonton zoning bylaws that are used in the prototype software application for checking:

```
public partial class CheckingForm : Form
{
    #region Properties

    private const double SingleDetached_MinSiteArea = 250.8;
    private const double SingleDetached_MinSiteWidth = 7.6;
    private const double SingleDetached_MinSiteDepth = 30;
    private const double Duplex_MinSiteArea = 300;
    private const double Duplex_MinSiteWidth = 10;
    private const double Duplex_MinSiteDepth = 30;
    private const double semiDetached_MinSiteArea = 488.4;
    private const double semiDetached_MinSiteWidth = 14.8;
    private const double semiDetached_MinSiteDepth = 30;
    private const double MaxBuildingHeight = 10;

    #endregion
}
```

In Edmonton city there are ten different zones (Zbylaw), that are Single Detached Residential Zone (RF1), Residential small Lot Zone (RSL), Low Density Infill Zone (RF2), Planned Lot Residential Zone (RPL), Small Scale Infill Development Zone (RF3), Semi-detached Residential Zone (RF4), Residential Mixed Dwelling Zone (RMD), Row Housing Zone, Urban Character Row Housing Zone (UCRH), Medium Density Multiple Family Zone (RF6). All the residential houses built in these zones are classified into single-detached housing (sdh), semi-detached housing (ssh), duplex housing (dh), limited group homes (lgh), garden suite (gs), secondary suites (ss), and minor home-based business (mhb).

Logical equations for the building regulations δ_{ss} , δ_{lot} , δ_{BC} , δ_{TH} and δ_{SB} are:

Check for lot dimensions:

$i = Z_{bylaw} = [\text{RF1, RF2, RPL, RF3, RF4, RMD, Row housing zone, UCHR, RF6}]$

$j = HT_{bylaw} = [\text{sdh, ssh, dh, lgh, gs, ss, mhb}]$

$$A = \begin{cases} \text{for } i \in 1 \dots 9 \\ \text{for } j \in 1 \dots 7 \\ M_{i,j} \leftarrow [i_j] \end{cases}$$

$$L_{bylaw} \in M_{9 \times 7} = A$$

$$\text{where, } A = \begin{pmatrix} \text{RF1}_{sdh} & \dots & \text{RF1}_{mhb} \\ \vdots & \ddots & \vdots \\ \text{RF6}_{sdh} & \dots & \text{RF6}_{mhb} \end{pmatrix}$$

get the user inputs:

$$\begin{cases} \sum Z_i = 1 (i) \\ \sum HT_j = 1 (j) \end{cases} \rightarrow \text{find } A(i, j) \rightarrow L_{bylaw(i,j)} = \begin{bmatrix} LA^{bylaw} \\ LD^{bylaw} \\ LW^{bylaw} \end{bmatrix}$$

$$\delta_{lot} = \begin{cases} 1 & \text{if } L_{bylaw(i,j)} > L_{BIM} \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation (1)}$$

Where :

Z_{bylaw} = Differnt types of residential zones present in Edmonton city.

HT_{bylaw} = Differnt types of residential buildings.

δ_{lot} = Check for the failure of lot dimensions.

$$L_{bylaw} = \begin{bmatrix} LA^{bylaw} \\ LD^{bylaw} \\ LW^{bylaw} \end{bmatrix} \quad L_{BIM} = \begin{bmatrix} LA^{BIM} \\ LD^{BIM} \\ LW^{BIM} \end{bmatrix}$$

L_{bylaw} = Lot dimensions information from bylaws data, i.e. LA^{bylaw} = Lot Area,
 LD^{bylaw} = Lot Depth, LW^{bylaw} = Lot Width.

L_{BIM} = Lot dimensions information from BIM model. LA^{BIM} = Lot Area,

LD^{BIM} = Lot Depth, LW^{BIM} = Lot Width.

Check for building coverage area:

$i = Z_{bylaw} = [\text{RF1, RF2, RPL, RF3, RF4, RMD, Row housing zone, UCHR, RF6}]$

$j = HT_{bylaw} = [\text{sdh, ssh, dh, lgh, gs, ss, mhb}]$

$k = LA = [\text{Area} < 300\text{sqm, Area} < 600\text{sqm, Area} \geq 600\text{sqm}]$

$$A = \begin{cases} \text{for } i \in 1 \dots 9 \\ \quad \text{for } j \in 1 \dots 7 \\ \quad \quad \text{for } k \in 1 \dots 3 \\ \quad \quad \quad M_{i,j} \leftarrow [i_j] \\ \quad MM_k \leftarrow M \end{cases}$$

$$BC_{bylaw} \in M_{9 \times 7 \times 3} = A$$

$$\text{where, } A = \begin{bmatrix} \text{RF1}_{sdh} & \dots & \text{RF1}_{mhb} \\ \vdots & \ddots & \vdots \\ \text{RF6}_{sdh} & \dots & \text{RF6}_{mhb} \end{bmatrix}$$

get the user inputs and BIM data:

$$\begin{cases} \sum Z_i = 1 (i) \\ \sum HT_j = 1 (j) \rightarrow \text{find } A(i, j, k) \rightarrow BC_{bylaw(i, j, k)} \\ LA_{BIM} = 1 (k) \end{cases} = \begin{bmatrix} PB_{bylaw} \\ AB_{bylaw} \\ PG_{bylaw} \end{bmatrix}$$

$$\delta_{BC} = \begin{cases} 1 & \text{if } BC_{bylaw(i, j, k)} < BC_{BIM} \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation (2)}$$

where:

Z_{bylaw} = Different types of residential zones present in Edmonton city.

HT_{bylaw} = Different types of residential buildings.

δ_{BC} = check for failure of building coverage percentage.

LA = Lot Area,

$$BC_{bylaw} = \begin{bmatrix} PB_{bylaw} \\ AB_{bylaw} \\ PG_{bylaw} \end{bmatrix} \quad BC_{BIM} = \begin{bmatrix} PB_{BIM} \\ AB_{BIM} \\ PG_{BIM} \end{bmatrix}$$

BC_{bylaw} = Building coverage area in accordance with Edmonton city bylaws.

BC_{BIM} = Building coverage area information from BIM model.

PB_{bylaw} = Principal building area percentage from bylaws.

AB_{bylaw} = Accessory building area percentage from bylaws.

PG_{bylaw} = Principal building attached with garage area from bylaws.

PB_{BIM} = Principal building area percentage compared with lot dimension information from BIM model.

AB_{BIM} = Accessory building area percentage compared with lot dimension information from BIM model.

PG_{BIM} = Principal building attached with garage area percentage compared with lot dimension information from BIM model.

Check for building height:

$i = Z_{bylaw} = [RF1, RF2, RPL, RF3, RF4, RMD, Row\ housing\ zone, UCHR, RF6]$

$j = HT_{bylaw} = [sdh, ssh, dh, lgh, gs, ss, mhb]$

$k = R = [Roof_{type1}, Roof_{type1}]$

$$A = \left\{ \begin{array}{l} \text{for } i \in 1 \dots 9 \\ \quad \text{for } j \in 1 \dots 7 \\ \quad \quad \text{for } k \in 1 \dots 2 \\ \quad \quad \quad M_{i,j} \leftarrow [i_j] \\ \quad \quad \quad MM_k \leftarrow M \end{array} \right.$$

$BC_{bylaw} \in M_{9 \times 7 \times 2} = A$

where, $A = \left\| \left\| \begin{array}{ccc} RF1_{sdh} & \dots & RF1_{mhb} \\ \vdots & \ddots & \vdots \\ RF6_{sdh} & \dots & RF6_{mhb} \end{array} \right\| \right\|$

get the user inputs and BIM data:

$$\begin{cases} \sum Z_i = 1 (i) \\ \sum HT_j = 1 (j) \rightarrow \text{find } A(i, j, k) \rightarrow TH_{bylaw(i, j, k)} \\ R_{BIM} = 1 (k) \end{cases}$$

$$\delta_{SC} = \begin{cases} 1 & \text{if } TH_{bylaw(i, j, k)} < TH_{BIM} \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation (3)}$$

where:

δ_{SC} = Check for failure of building height.

R= Roof types, $Roof_{type1} \in \{\text{Flat, Sloped}\}$. $Roof_{type1} \in \{\text{Gable, Hip, Mansard, Gambrel, Flat}\}$

R_{BIM} = Roof type from BIM model.

BH_{bylaw} = Building height permitted.

TH_{BIM} = Total height of building information from BIM model.

TH_{bylaw} = Total height of building permitted in accordance with bylaws.

Z_{bylaw} = Differnt types of residential zones present in Edmonton city.

HT_{bylaw} = Differnt types of residential buildings.

Check for setback distance:

$i = Z_{bylaw} = [\text{RF1, RF2, RPL, RF3, RF4, RMD, Row housing zone, UCHR, RF6}]$

$j = HT_{bylaw} = [\text{sdh, ssh, dh, lgh, gs, ss, mhb}]$

$k = C = [C_1, C_2, C_3]$

$$A = \begin{cases} \text{for } i \in 1 \dots 9 \\ \quad \text{for } j \in 1 \dots 7 \\ \quad \quad \text{for } k \in 1 \dots 3 \\ \quad \quad \quad M_{i,j} \leftarrow [i_j] \\ \quad \quad \quad MM_k \leftarrow M \end{cases}$$

$$BC_{bylaw} \in M_{9 \times 7 \times 3} = A$$

$$\text{where, } A = \begin{bmatrix} RF1_{sdh} & \dots & RF1_{mhb} \\ \vdots & \ddots & \vdots \\ RF6_{sdh} & \dots & RF6_{mhb} \end{bmatrix}$$

get the user inputs and BIM data:

$$\begin{cases} \sum Z_i = 1 (i) \\ \sum HT_j = 1 (j) \rightarrow \text{find } A(i, j, k) \rightarrow SB_{bylaw(i, j, k)} \\ \sum C_k = 1 (k) \end{cases}$$

$$\delta_{SB} = \begin{cases} 1 & \text{if } (SB_{bylaw}^{Min.} > SB_{BIM}) \\ 1 & \text{if } (SB_{bylaw}^{Max.} < SB_{BIM}) \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation (4)}$$

Where:

δ_{SB} = Check for failure of setback distances of buildings.

C = different building conditions, $C_1 \in \{\text{Corner site}\}$,

$C_2 \in \{\text{Corner site, Attached garage}\}$,

$C_3 \in \{\text{Corner site, Attached garage, Building faces flankside}\}$

$$SB_{bylaw}^{Min.} = \begin{bmatrix} FSB_{bylaw}^{Min.} \\ SSB_{bylaw}^{Min.} \\ RSB_{bylaw}^{Min.} \end{bmatrix} \quad SB_{bylaw}^{Max.} = \begin{bmatrix} FSB_{bylaw}^{Max.} \\ SSB_{bylaw}^{Max.} \\ RSB_{bylaw}^{Max.} \end{bmatrix} \quad SB_{BIM} = \begin{bmatrix} FSB_{BIM} \\ SSB_{BIM} \\ RSB_{BIM} \end{bmatrix}$$

$SB_{bylaw}^{Min.}$ = Minimum setback distance permitted according to bylaws.

$SB_{bylaw}^{Max.}$ = Maximum setback distance permitted according to bylaws.

SB_{BIM} = Setback distances information from BIM model.

FSB_{bylaw} = Front setback distance (minimm and maximum).

SSB_{bylaw} = Side setback distances (minimm and maximum).

RSB_{bylaw} = Rear setback distance (minimm and maximum).

FSB_{bylaw} = Front setback infromation from BIM model.

SSB_{bylaw} = Side setback information from BIM model.

RSB_{bylaw} = Rear setback information from BIM model.

Check for stud spacing:

$$i = S_{size} = [2 \times 2, 2 \times 4, 2 \times 6, \dots, m \times n]$$

$$j = S_c = [C_1, C_2, C_3, C_4, C_5, C_6]$$

$$k = T_{wall} = [\text{interior}, \text{exterior}]$$

$$A = \begin{cases} \text{for } i \in 1..n \\ \quad \text{for } j \in 1..6 \\ \quad \quad \text{for } k \in 1..2 \\ \quad \quad \quad M_{i,j} \leftarrow [i_j] \\ \quad \quad \quad MM_k \leftarrow M \end{cases}$$

$$BC_{bylaw} \in M_{n \times 6 \times 2} = A$$

$$\text{where, } A = \begin{bmatrix} 2 \times 2_{C_1} & \dots & 2 \times 2_{C_6} \\ \vdots & \ddots & \vdots \\ mxn_{C_1} & \dots & mxn_{C_6} \end{bmatrix}$$

get the stud, load and wall type details from BIM model:

$$\begin{cases} \sum S_{size} = 1 (i) \\ \sum S_c = 1 (j) \rightarrow \text{find } A(i, j, k) \rightarrow SS_{bylaw(i, j, k)} \\ \sum T_{wall} = 1 (k) \end{cases}$$

$$\delta_{SS} = \begin{cases} 1 & \text{if } (SS_{bc} < SS_{BIM}) \\ 0 & \text{otherwise} \end{cases} \quad \text{Equation (5)}$$

Where:

S_{size} = stud dimensions.

S_c = wall support conditions.

C_1 = {no load}, C_2 = {Attic not accessible by a stairway},

C_3 = {Attic Accessible by a stairway plus one/two floor, roof load plus one floor}

C_4 = {roof load, Attic not accessible by a stairway,

Attic not accessible by a stairway plus one floor}

$C_5 = \{\text{Attic accessible by a stairway plus two floors, Roof load plus two floors}\}$

$C_6 = \{\text{Attic accessible by a stairway plus 3 floors, Roof load plus two floors}\}$

$T_{wall} = \text{wall type.}$

$\delta_{SS} = \text{Check for failure of stud spacing.}$

$SS_{bylaw} = \text{maximum stud spacing according to building code.}$

$SS_{BIM} = \text{stud spacing details from BIM model.}$

3.1.2 BIM model preparation

The BIM model can be defined as a digital representation of physical characteristics like architectural designs or construction drawings, and functional characteristics like structural analysis, energy analysis, or a myriad of other simulations with semantically rich information (Chuck Eastman, 2009), which has been a revolutionary technology in the AEC industry. Designing an enriched BIM model in Revit is the primary requirement, where the object-based information modelling would contain the required level of detail for comprehensive automated code compliance. Building objects modelled in any BIM-enabled software will have parametric data and properties. For example, a wood stud member in the model will possess types and properties like dimensions (length, width, depth), material properties, location of element member (XYZ coordinates), and so on. Because of this, the models developed for code compliance must be accurate with certain details provided for all the objects modelled. For this automated design checking, the building model can be developed with level of details (LOD) above 300, and in this case, the BIM model will contain the required details of building objects with quantity, size, location, and systematic relationships of object with related to other building elements. After developing the BIM model with the level of details above 300, the required information L_{BIM} , BC_{BIM} , TH_{BIM} , SB_{BIM} and SS_{BIM} from the model for code compliance

checking can be extracted without much need to build the extended data structure to derive information and get information from users as inputs. Revit provides a way to define and export extensible and interoperable BIM model data with use of Revit's API (application program interface). The API provides information about building objects designed in the model, by using appropriate API information related to L_{BIM} , BC_{BIM} , TH_{BIM} , SB_{BIM} and SS_{BIM} , objects can be extracted. The structure of APIs for exchanging information is object-based, where the geometry and properties of objects, such as name, size, location, finishes, faces, and abstract information like cost, quantities, and so on can be accessed. C# language in Visual Studio Express 2015 has been used to build the data structure platform called DCheck for exchanging information for compliance checking, and is used for extracting to L_{BIM} , BC_{BIM} , TH_{BIM} , SB_{BIM} and SS_{BIM} information from the BIM model. As such, for getting details about "property line" to check for lot dimensions, "PropertyLine" is filtered from the BIM document, storing all the BIM model information about this in a list and accessing necessary information when required.

```
FilteredElementCollector(_doc).OfClass(typeof(PropertyLine)).ToElements().ToList().
```

Some of the properties can be accessed easily with a single command, like for getting the area of "property line" by using `propertyLine.get_BoundingBox(_doc.ActiveView)`. Some information like building area compared with lot area in percentage must be derived where we cannot access the required details directly. DCheck platform has the data structure for checking building rules and regulations related to wall framing and zoning bylaws mentioned in Table 4.

3.1.3 Rule Checking

The main fundamental aspect of automated compliance checking is the exchange of information from BIM model with the database platform. However, the building information present in the BIM model itself is not adequate for compliance checking. This is because some of the rules to be satisfied, such as check for stud maximum spacing, minimum setback

distances, building coverage area, and so on, require building details that cannot be directly accessed from the BIM model. In this case, we would require higher level semantics of building elements that can be derived from BIM model developed with LOD more than 300. In the case for automated framing and zoning bylaw checking, these are provided by the DCheck platform. With object-based building information extraction, the geometric topology and functional information of objects, such as faces of building components, vertices, edges, location, and some derived information that are required for compliance checking are retrieved by this DCheck system. Mapping between the BIM model and building rules will be done in DCheck platform in this step, as shown in Figure 3.1. DCheck platform has been developed to support and to get extended information about objects, and this platform is designed to be extendable for customization to deal with different conditions, and for updates of regulations. An example of how DCheck functions are utilized in rules conformance verification are described here with site coverage area, that is, the percentage of area of the site that is covered by the building at ground level. A DCheck class `GetMaxCoverage` is invoked for the facility to check for site coverage area, which includes various methods to perform checking with different conditions. Some of the methods used are `buildingCoverage`, which retrieves the building coverage area; and `Total_Site_Coverage`, which retrieves the total site coverage area. Every time the user runs checking process, DCheck platform will access BIM model data and compute threshold values retrieved from the model with building regulation values according to rules.

3.1.4 Checking Report

The final step of the automated compliance checking process is providing the user with a final compliance checking report by notifying where checking results are a success or failure, and giving suggestions for failed regulation with a reason for failure. This report is displayed for the user in textual format, and those building objects related to the rules which they have failed to satisfy will be highlighted in the model, which helps to spot those objects and make

corrections. The checking process can be run at any time during the design by the user, so it is easy for the architects or the draftspersons to check models in a parallel manner while designing, which helps make the approval process for construction easy and more efficient.

3.2 Prototype Development

The automated building design checking is implemented as an add-on software application for the Autodesk Revit software, developed in C# language using Revit API's for retrieving L_{BIM} , BC_{BIM} , TH_{BIM} , SB_{BIM} and SS_{BIM} information from the BIM model. Autodesk Revit is a powerful modelling tool and is open source, which allows the user to develop extension applications if required to perform more advanced operations. Figure 3.2 shows the architecture of the prototyped Revit-based automated design checking software application. The inputs for the system include: (1) building design of the project and BIM model of the building intended to be constructed containing the architectural and structural framing information modelled with a certain level of details; (2) the project applicant information, regarding the applicant, architect and builder information, and (3) zoning and framing information, regarding site location, plot number, type of wood used for framing and so on. Criteria for this project are as follows: (1) building code, in this case Alberta Building Code 2014 Part 9, housing and small buildings, containing regulations related to framing of residential building; (2) municipal bylaws, Edmonton zoning bylaws, containing regulations related to different types of zones and residential building, and (3) Level of details (LOD), development of model with LOD above 300 will serve the required purpose for automation of building rules checking in this prototype.

The core processor of this prototype as shown in Figure 3.2 has main components: (1) object-based representation of building code and model, where building rules are represented based on the building objects so that required information from building model related to particular

building object can be known clearly for extracting information; (2) BIM model view definition, where required model views information from BIM data for compliance checking will be extracted; (3) BIM model extension, where some information from the BIM model needs to be derived which cannot be accessed directly so DCheck extended data structure platform will provide those values; and (4) code compliance with model, where extracted information from model will be checked against the already-encoded building rules. These four components are compiled into Autodesk Revit as an add-on through using C# language. The object-oriented data of BIM model, in which building objects will have enriched properties from which geometrical, topology, functional information of building objects can be extracted efficiently. For compliance checking, data from BIM model will be extracted in accordance with objects attribute values, which may be material type, geometric placement reference to object or x, y, z point locations. The output of this process involves displaying of final checking result in textual format informing whether compliance checking is successful or has failed, and if failed, displaying the failed rules with suggestions to correct those failed rules and objects related to those rules can be visually represented by automatically highlighting the objects related to failed rules.

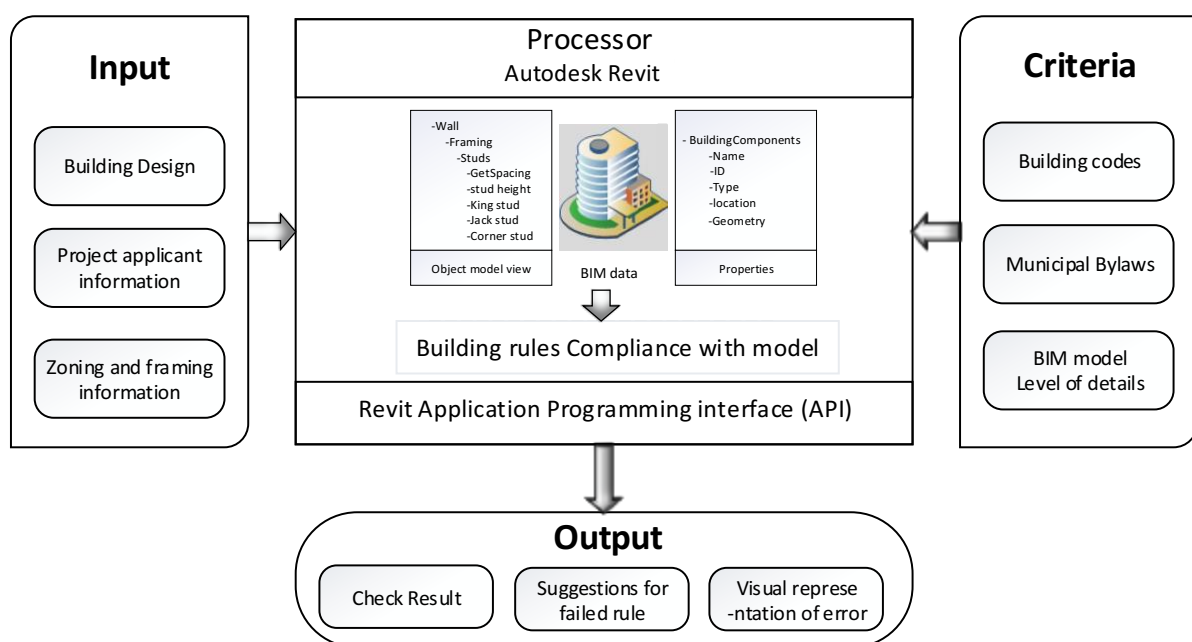


Figure 3. 2. System Architecture.

All the building rules are classified into three different types based on the complexity in interpreting natural language rules into machine-readable format and extraction of information from BIM model for compliance. These three types are as follows:

(1) Rules that are classified as easy to translate and where the information required can be directly extracted from model. These are rules which can be translated from natural human-readable language into C# easily, for example “In Single Detached Residential Zone (RF1), types of houses that are allowed to be constructed are single detached housing, secondary suites, semi-detached housing, duplex housing”. This kind of rules can be translated to C# with simple functional conditions. And the information required for checking related to this kind of rules, i.e., types of zone and house, can be directly accessed from the BIM model, where in this case the user will be providing that information related to types of zones and houses before checking.

(2) Rules that are classified as difficult to translate and where the information required needs to be derived. The complexity level of translating these rules into C# language and getting that information from BIM model involves introducing some new attribute values for defining some properties of building objects for compliance. For example, “In corner site where the building faces the front lot line or the side lot line, the minimum side setback abutting the flanking side lot line shall be 20% of the site width, to maximum of 4.5 m.”. In these kinds of rules, it is bit difficult to translate to C#. First, it has to satisfy several conditions to check for threshold value related to side setback distance. And then, to get this required information from the BIM model, some new attributes should be used for deriving side setback distance for checking because we cannot access required values directly by using APIs.

(3) Rules that are classified as needing to be simplified in order to translate them, and where

the information required needs extended data structure. Some rules need clear understanding depending on the building design and specifications, and some rules need the building model to be analyzed to get the required value to be checked, and the information required from the BIM model will be extracted using extended data structure. For example, “A secondary suite shall be developed in such a manner that the exterior of the principal building containing the secondary suite shall appear as a single dwelling”. In this case, the characteristics of the single dwelling in exterior appearance have to be considered, and checking the model against this kind of rule requires that all the characteristics of single dwellings to be encoded in checking platform.

3.2.1 Edmonton zoning bylaw checking

Automation of building rules compliance checking process will make the construction permit approval process easier and help get the construction process started sooner. With the automation of compliance checking process and the effective use of the application by both designer and approval authorities, there will not be any delays in approval of drawings for construction. Before construction of any houses in the city of Edmonton, the design drawings have to be checked by the authorities responsible for approval for construction in accordance with Edmonton Zoning Bylaws. In this study, we are automating the regulations related to lot checking for different types of houses constructed in different zones. The implementation process is explained here with some of the rules included in this study. According to Edmonton city bylaws, principal and accessory buildings can be constructed inside the boundaries of the site, which will have minimum offsets from building or facilities that exist in site. The boundary of site is known as property line. The details related to the building are designed with certain level of details by architects in Autodesk Revit and the information relating to lot shape, site characteristics, building design and specifications of building that are required to be submitted for approval for construction, has to be modelled and the user should provide

initial inputs regarding zoning and housing type. This is same information the user used to have to provide with submission of 2D drawings. In the city of Edmonton, there are ten different zones: Single Detached Residential Zone (RF1), Residential small Lot Zone (RSL), Low Density Infill Zone (RF2), Planned Lot Residential Zone (RPL), Small Scale Infill Development Zone (RF3), Semi-detached Residential Zone (RF4), Residential Mixed Dwelling Zone (RMD), Row Housing Zone, Urban Character Row Housing Zone (UCRH), Medium Density Multiple Family Zone (RF6). All the residential houses built in these zones are classified into single-detached housing, semi-detached housing, duplex housing, limited group homes, garden suite, secondary suites, and minor home-based business. Depending on the different type of zone, there are different conditions to be satisfied for building a specific permitted type of house, as shown in Figure 3.3 below, which also gives details about the minimum setback distances that should be provided depending on zone type, dwelling type and built with or without attached garage or just principal dwelling. Below is an example of type 1 rules, where the information required for checking for those building rules can be easily obtained from BIM model.

“If it is Single Detached Housing: Minimum site/ Lot Dimensions should be, Area: 250.8 m². Width: 7.6 m². Depth: 30 m²”

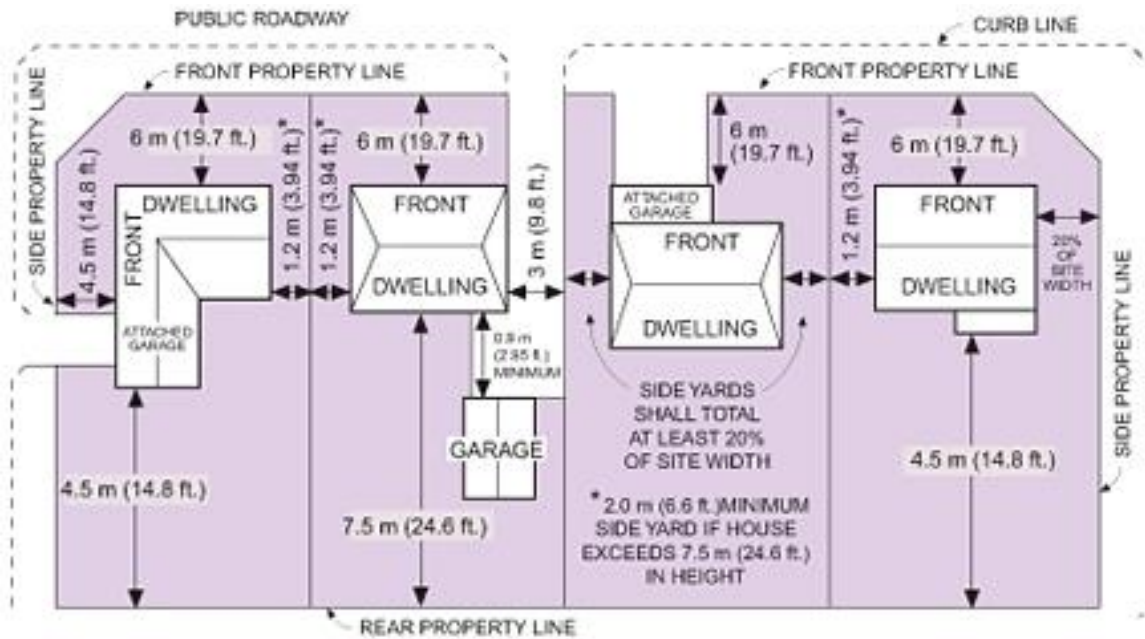


Figure 3.3. Illustration of Setback Distances Requirements for RE1 Zone (Adopted from City of Edmonton Website)

The generalized flow chart of the checking process for the lot dimensions is shown in Figure 3.4, which checks with specified zone and house type based on data given by the user. Each type of house has different lot dimension properties conditions to be satisfied in different zones. Based on user inputs, threshold values related to minimum area of lot (A_{lot}), minimum width of lot (W_{lot}), and minimum depth of lot (D_{lot}) to be checked are chosen from the municipal bylaw data, which are the threshold values that can be easily changed if required for checking different jurisdictions' regulations. Those values are then checked against designed building model lot dimensions. And if the designed model failed to satisfy any of the rules, then that particular rule which failed to be satisfied will be displayed after the complete check for all the rules encoded in the add-on software application is done. Results will be in text format, where the building object related to that failure will be highlighted and reasons for the failure with suggestions to correct for that error will be displayed, and from there the user can easily identify and make changes as required and run the checking add-on software application again to see if the design is successfully checked without any errors.

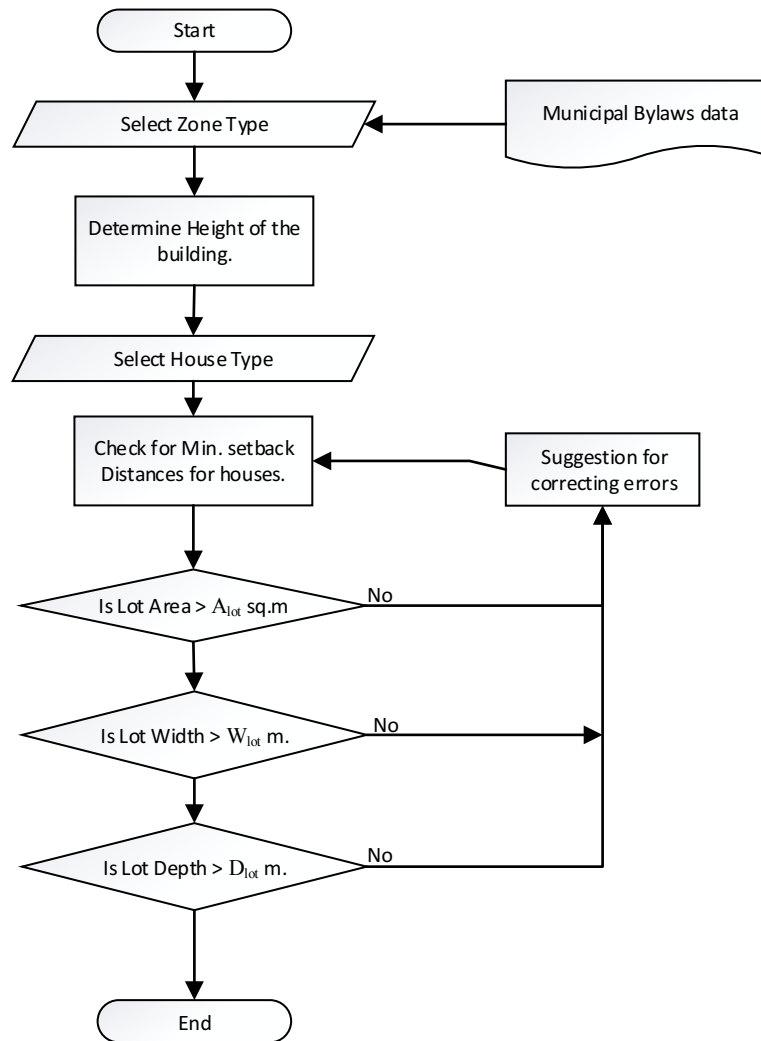


Figure 3. 4. Flow Chart for checking Lot Dimensions.

With representation of the building rules in logical form, it will be more convenient for understanding rules for better interpretation. First order logic (FOL) is also known as predicate logic, which uses quantified variables over non-logical objects and allows the use of sentences that contain variables. FOL is also is a symbolized reasoning in which each sentence is broken down into a subject and predicate. That predicate will define the properties of subject. In first-order logic (FOL), the above rule for lot dimension checking can be expressed as:

$$\begin{aligned} & \forall a \forall b (\text{Zone}(a) \wedge \text{House}(b) \wedge \text{Type}(b, a)) \\ & \wedge \exists b ((\text{House}(b) \wedge \text{Check}(b, A_{\text{lot}}) \wedge (\text{House}(b) \wedge \text{Check}(b, W_{\text{lot}}) \wedge (\text{House}(b) \wedge \text{Check}(b, \\ & D_{\text{lot}})) \\ & \wedge \forall b ((\text{House}(b) \wedge \text{Check}(b, H_{\text{house}})) \end{aligned}$$

where

$a \in \{\text{RF1, RSL, RF2, RPL, RF3, RF4, RMD, Row Housing Zone, UCRH, RF6}\}$

$b \in \{\text{Single Detached Housing, Semi-Detached Housing, Duplex Housing, Limited group homes, Garden suite, Secondary suites, Miner Home Based Business}\}$

$A_{\text{lot}} = \text{Lot Area.}$

$W_{\text{lot}} = \text{Lot Width.}$

$D_{\text{lot}} = \text{Lot Depth.}$

$H_{\text{house}} = \text{height of house.}$

As the first-order logic equation represents checking for lot dimensions for all the zones and house, only some house types can be built in a particular zone, and depending on that, the check for house minimum lot area, width and depth is made. If the regulation may be represented in FOL, then it will be easy for interpreting into machine-readable format by referring to it. These types of rules can be easily coded into C# and the information required can be extracted directly from BIM model by using simple Revit API commands. The property line is required to get details related to area, width and depth of site, and can be accessed by knowing the boundary details like `propertyLine.get_BoundingBox(_doc.ActiveView)`.

Figure 3.5 shows the lot dimensions checking for different type of houses based on Edmonton municipal zoning bylaws with specific threshold values to be checked for successful lot dimensions compliance.

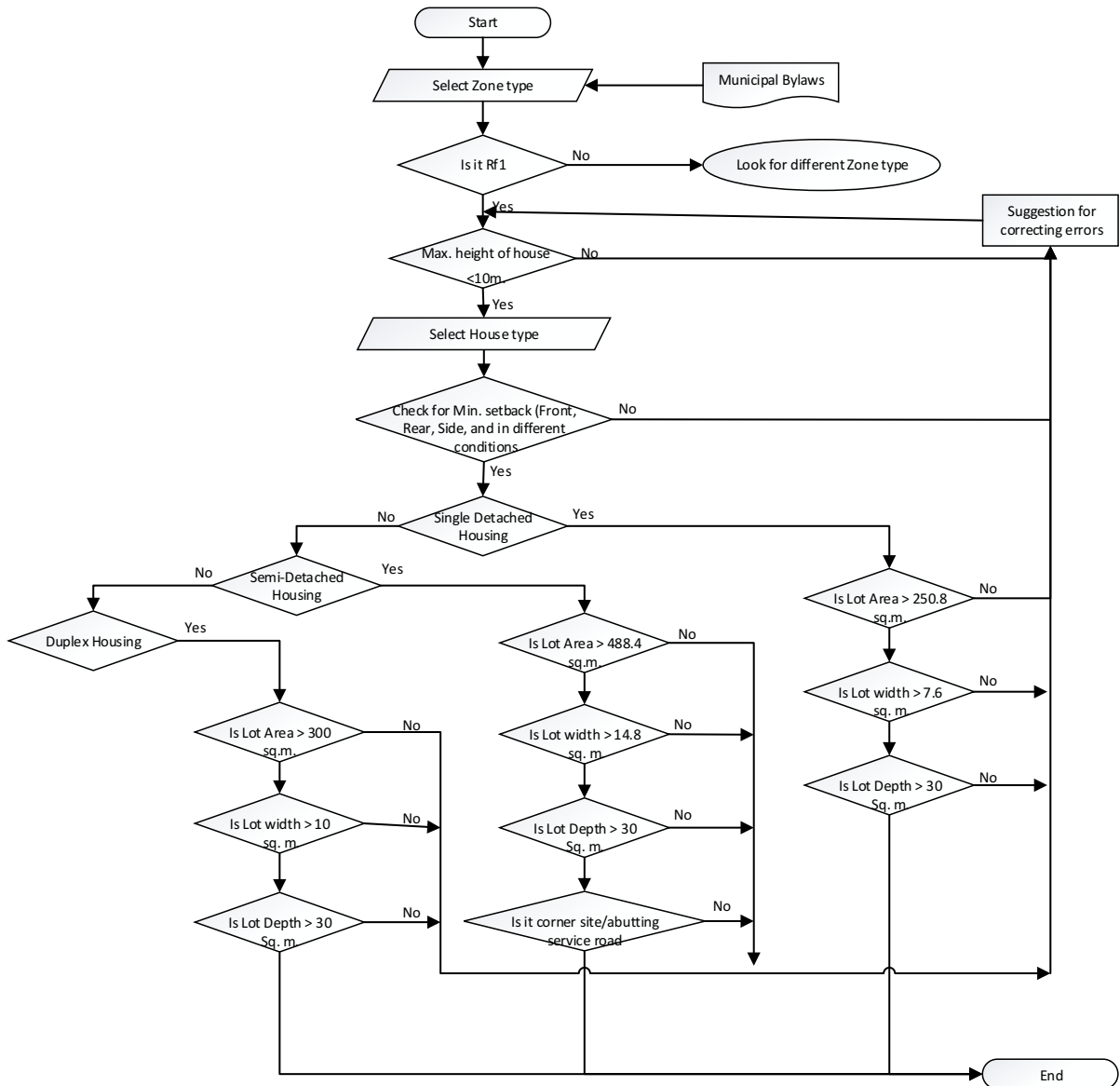


Figure 3. 5. Flow Chart for Checking Lot Dimensions of Different Types of Houses.

Figure 3.6 shows the checking flow chart for checking percentage of lot area to be covered by the house at ground level. This type of rule can be an example for type 2 rule classification, where it will be a bit difficult to code this rule into $C\#$, and where the information required for satisfying the conditions required has to be derived, i.e., the value to be checked cannot be directly obtained from BIM model as in type 1 classification. Below is the table representing site coverage rules with respect to specific housing type and area to be satisfied according to City of Edmonton bylaws.

Table 2. Maximum Site Coverage with Respect to Specific Housing Type and Area.

	Principal Dwelling/Building	Accessory building	Principal Building with Attached Garage
Single Detached Housing – Site area greater than 300 m ²	28%	12%	40%
Single Detached Housing – Site area less than 300 m ²	28%	14%	42%
Duplex Housing	28%	12%	40%
Semi-detached Housing- Site area 600 m ² or greater	28%	12%	40%
Semi-detached Housing- Site area less than 600 m ²	28%	14%	42%
All other Uses	28%	12%	40%

As shown in Table 3, depending on site area, house type, and whether the garage is attached or not, the percent of area the house should cover on the site has to be checked. The house area can be obtained by getting dimensional measurements of foundation walls by grouping them all, and the area of the site can be obtained directly from BIM model, and getting the percentage of area covered depends on the user-specified house type and comparing the BIM model value with values in the above table.

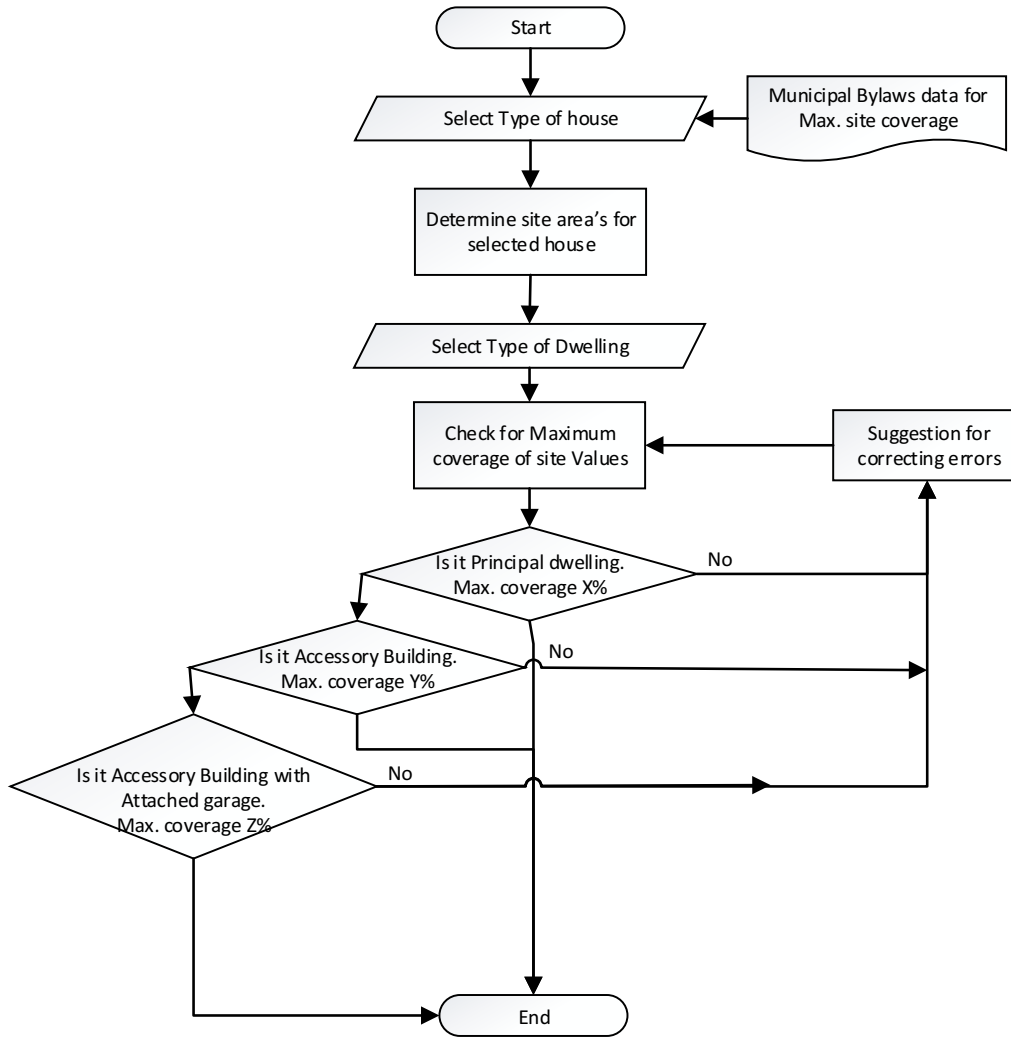


Figure 3. 6. Flow Chart for Checking House Area.

In first-order logic (FOL), the above rules for checking house area are shown as:

$$\begin{aligned}
 & \forall a \forall b (\text{House}(a) \wedge \text{DwellingType}(b) \wedge \text{Type}(b, a)) \\
 & \wedge \exists b ((\text{DwellingType}(b) \wedge \text{Check}(b, A_{\text{house}}) \vee (\text{DwellingType}(b) \wedge \text{Check}(b, A_{\text{accessory}}) \vee \\
 & (\text{DwellingType}(b) \wedge \text{Check}(b, A_{\text{Principal}}))
 \end{aligned}$$

where

$a \in \{\text{Single Detached Housing, Semi-Detached Housing, Duplex Housing, Limited group homes, Garden suite, Secondary suites, Miner Home Based Business}\}$

$b \in \{\text{Principal dwelling, Accessory Building, Principal Building with attached Garage}\}$

A_{house} = Area of Principal dwelling.

$A_{\text{accessory}}$ = Area of Accessory building.

$A_{\text{principal}}$ = Area of Principal building with attached garage.

Depending on different housing type in RF1 zone, the flow chart for checking building coverage area depending on different type of houses and type of building in accordance with Edmonton municipal zoning bylaws is shown in Figure 3.7 below.

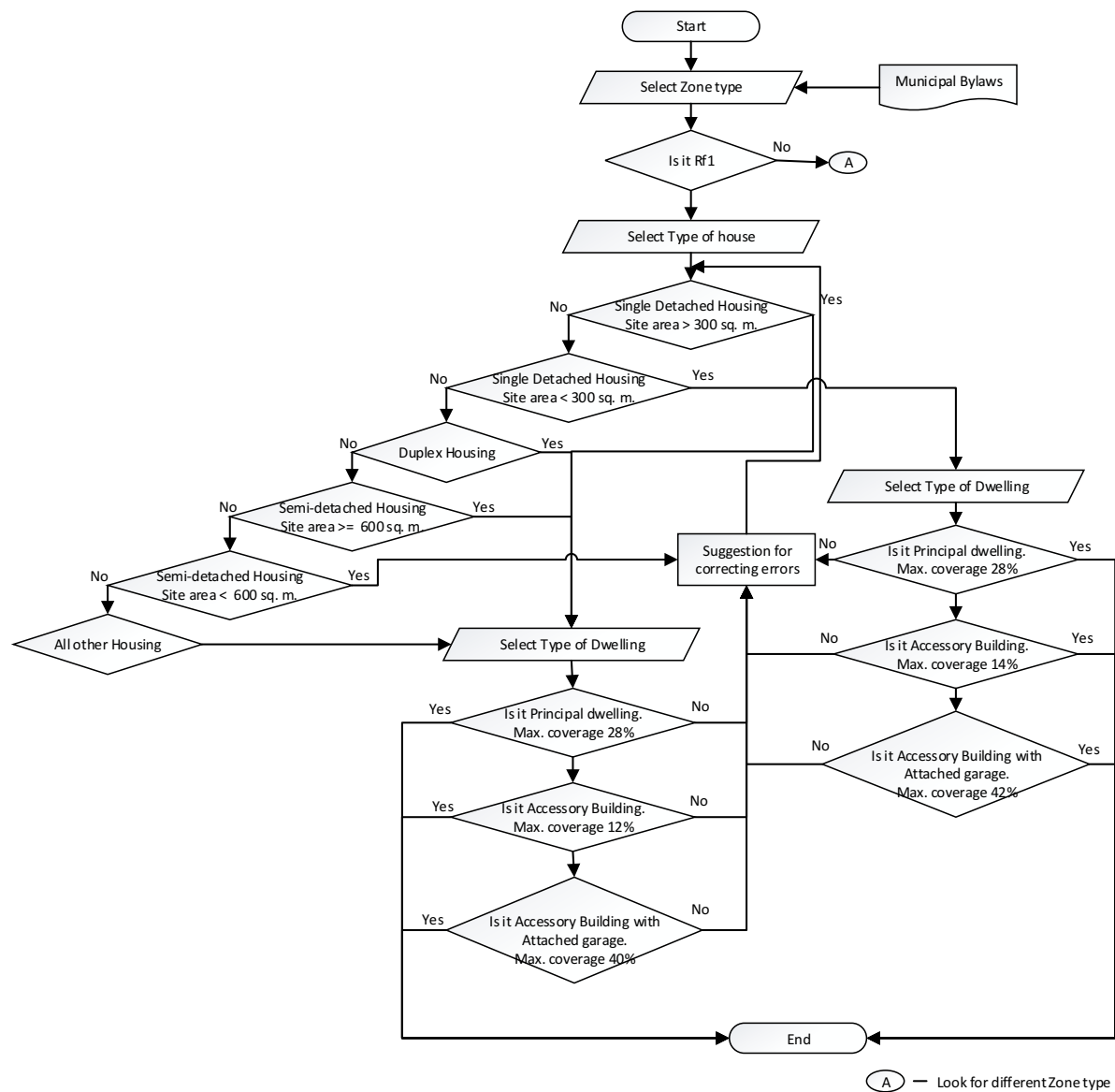


Figure 3. 7. Flow Chart for Checking of Building Coverage for Different Types of Houses.

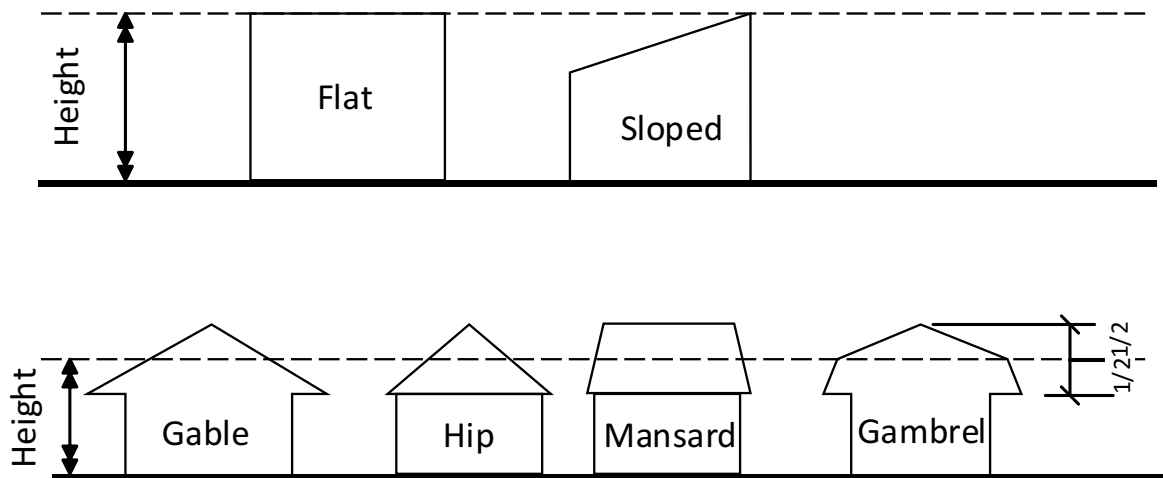


Figure 3. 8. Height Consideration for Types of Houses (Adopted from City of Edmonton).

The height of house is measured differently according to specific housing roof design as shown in Figure 3.8 below. This type of rule can be an example for type 3 classification of rules, where we need to use some derived attribute values for checking type of roof and determine height based on that criteria. According to the City of Edmonton bylaws, the height of any kind of house in the single detached residential zone (RF1) cannot exceed 10 m (32.8 ft) or 2.5 storeys. First, the face details of roof are used to check whether it is flat or sloped by defining the conditions for flat and sloped roof, and check for those conditions and if it does not satisfy flat or sloped roof condition then it can be considered as gable or hip or mansard or gambrel roof. Figure 3.9 shows the flow chart for the height calculation. If the roof type is flat or sloped, total height of house must be considered as top floor wall height plus roof height, or if the roof is of gable or hip or mansard or gambrel type, then total height of house should be considered as top floor wall height plus half of the roof height.

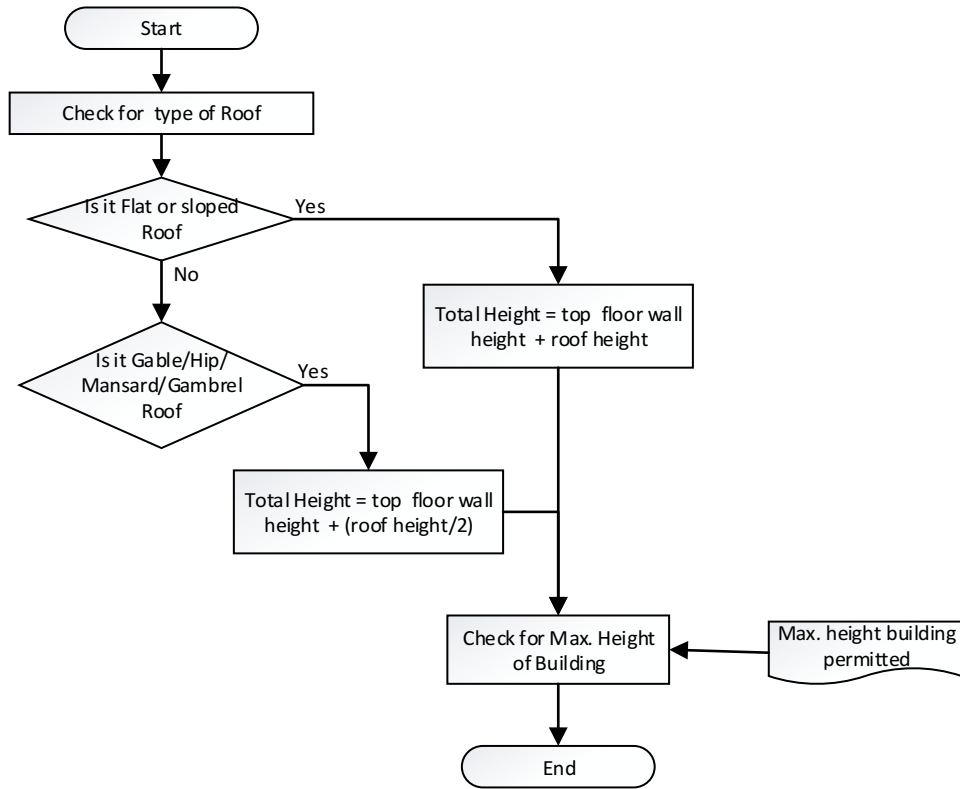


Figure 3. 9. Flowchart for checking Height of Building.

In FOL, to determine the height of house can be shown as:

$$\begin{aligned}
 & \forall a \forall b (\text{House}(a) \wedge \text{Roof}(b) \wedge \text{has}(a, b)) \\
 & \wedge \exists b ((\text{Roof}(b) \wedge \text{Check}(b, R_{\text{flat/sloped}}) \wedge \text{Get}(b, h_{\text{roof}}) \vee (\text{Roof}(b) \wedge \text{Check}(b, R_{\text{Gable/Hip/Mansard/Gambrel}}) \wedge \text{Get}(b, h_{\text{roof}}))) \\
 & \wedge \forall a \exists b ((\text{House}(a) \wedge \text{Check}(a, H_{\text{height}})
 \end{aligned}$$

where

$a \in \{\text{Single Detached Housing, Semi-Detached Housing, Duplex Housing, Limited group homes, Garden suite, Secondary suites, Miner Home Based Business}\}$

$b \in \{\text{flat, Sloped, Gable, Hip, Mansard, Gambrel}\}$

$R_{\text{flat/sloped}} = \text{Flat or Sloped roof type.}$

$R_{\text{Gable/Hip/Mansard/Gambrel}} = \text{Gable/Hip/Mansard/Gambrel roof type.}$

$H_{\text{roof}} = \text{Height of the roof.}$

H_{height} = Height of the building.

3.2.2 Framing Checking

Another aspect of building design that has been included in this research is wood framing checking for residential buildings. Wood framing construction comprises main structural members (the framing) and sheathing (orient strand board or plywood that provides stiffness). The combination of framing members and sheathing provides rigidity, spaces for insulation, and a framework for supporting interior finishes and exterior components. Framing works in conjunction with the house's foundation to provide strength and stability for the structure by transferring load to the foundation (Practices, n.d.). Residential buildings built in Alberta must follow the Alberta Building Code (ABC) 2014, part 9 for framing design details of the building. All of the residential buildings are framed in accordance with the ABC, which is adapted from National Building Code. Only in some unique, critical or exceptional cases, the structural designer needs to design the structure as per bylaws, apart from that all framing of residential buildings are designed using building code. So, automating the checking process of framing according to building code helps in validating the model very fast, with consistency and in less time. Checking these kind of design rules can be done once the design of the framing of the building is done, and by using the same add-on software application, the user can provide the inputs required for checking framing. Below is a description of some building code examples related to framing checking that have been implemented in the prototype add-on software application. Figure 3.10 shows the flowchart of some rules check for framing related to spacing, height with different stud dimensions. These kinds of rules can be classified as type 2, where we need to derive some values from the BIM model to check these codes. Table 4 below shows some of the building code related to wall framing spacing and maximum height of studs based on different support conditions and stud dimensions. First, the load conditions, type of wall and wall support information are taken from the BIM model, which

are provided by the user at the

beginning of the checking process. The spacing and maximum height for stud dimensions used in framing are selected from data and checked against derived BIM model values specific to Alberta Building Code is as shown in Figure 3.11.

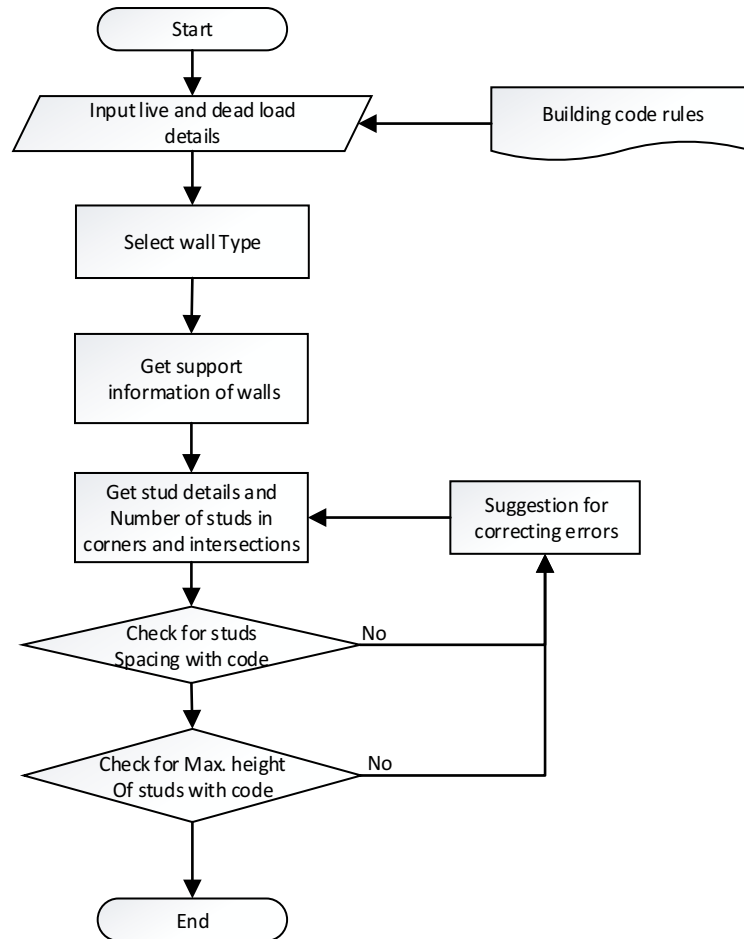


Figure 3. 10. Flow Chart for Framing Checking for Residential Building.

In FOL, the above flowchart for framing checking with few rules can be shown as:

$$\begin{aligned}
 & \forall a \forall b (\forall s(\text{House}(a) \wedge \text{Wall}(b) \wedge \text{Studs}(s) \wedge \text{different}(a, b) \wedge \text{Structuralmember}(s, b)) \\
 & \wedge \exists l \forall d \forall e (\exists c(\text{Studdimension}(c) \wedge \text{Spacing}(d) \wedge \text{Load}(l) \wedge \text{Height}(e) \wedge \text{Check}(s, d) \wedge \text{Check}(s, e)) \\
 & \wedge \forall f \forall g(\text{Studcount}(g) \wedge \text{Corners\&intersections}(f) \wedge \text{Check}(f, g)))
 \end{aligned}$$

where

a ∈ {Single Detached Housing, Semi-Detached Housing, Duplex Housing, Limited group homes, Garden suite, Secondary suites, Miner Home Based Business}

b ∈ {Interior, Exterior}.

Table 3. Spacing and Maximum Height of Studs.

Type of wall	Supported loads (including dead loads)	Minimum Stud Size, mm	Maximum Stud Spacing, mm	Maximum Unsupported Height, m
Interior	No Load	38×38	400	2.4
		38×89 flat	400	3.6
	Attic not accessible by a stairway	38×64	600	3.0
		38×64 flat	400	2.4
		38×89	600	3.6
		38×89 flat	400	2.4
	Attic Accessible by a stairway plus one floor Roof load plus one floor Attic not accessible by a stairway plus 2 floors	38×89	400	3.6
	Roof load Attic accessible by a stairway Attic not accessible by a stairway plus one floor	38×64	400	2.4
		38×89	600	3.6
	Attic accessible by a stairway plus 2 floors Roof load plus 2 floors	38×89	300	3.6
64×89		400	3.6	
38×140		400	4.2	
Attic accessible by a stairway plus floors Roof load plus 2 floors	38×140	300	4.2	
Exterior	Roof with or without attic storage	38×64	400	2.4
		38×89	600	3.0
	Roof with or without attic storage plus one floor	38×89	400	3.0
		38×140	600	3.0
	Roof with or without attic storage plus 2 floors	38×89	300	3.0
		64×89	400	3.0
		38×140	400	3.6
	Roof with or without attic storage plus 3 floors	38×140	300	1.8

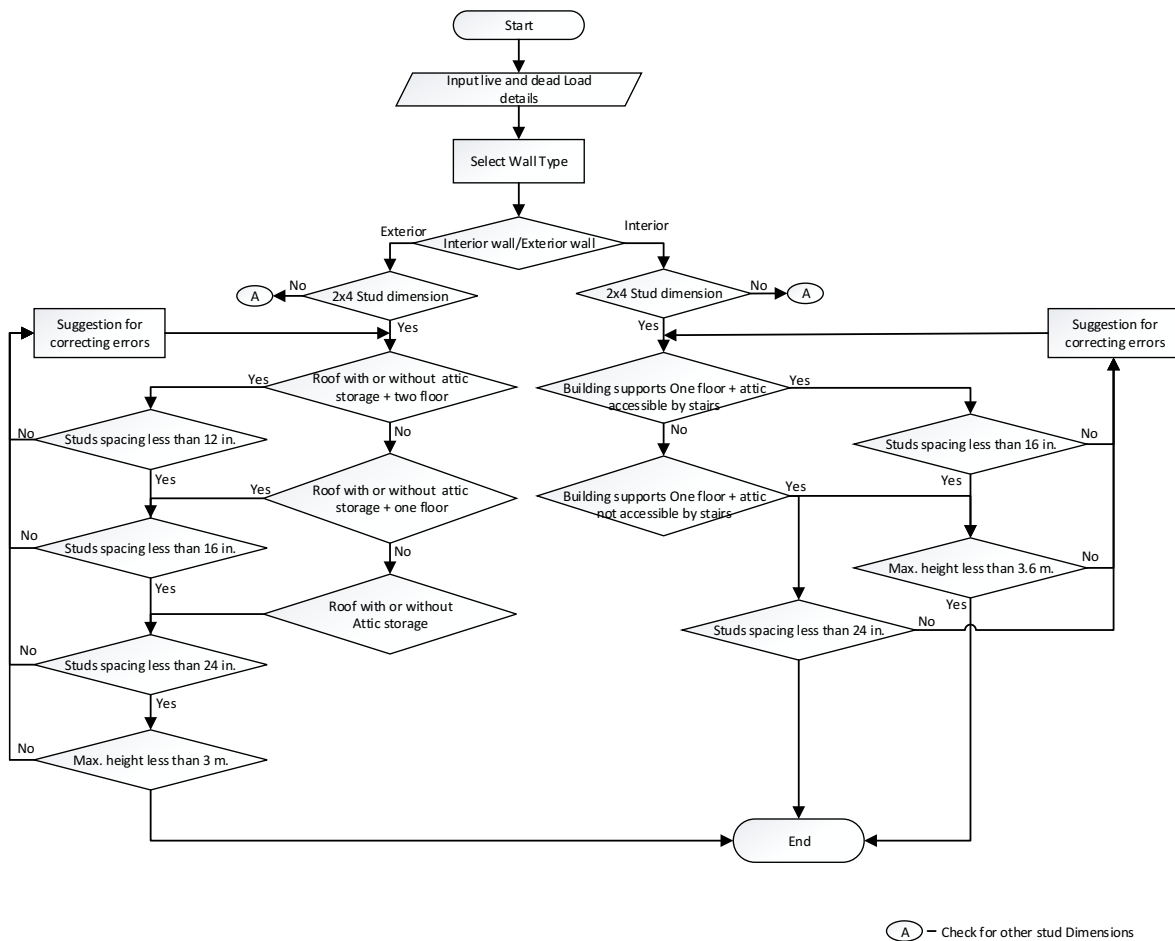


Figure 3. 11. Flow Chart for Checking Spacing and Maximum Height of Framing.

3.2.3 Domain knowledge from the standard

An object-oriented representation of the bylaws and code with a threshold value to be checked along with the conditions to be satisfied is the best way of representing the code for interpreting those rules into C# coding language, where the building designs will be modelled in Autodesk Revit and are always fully coordinated in terms of the building objects. Add-on software application for Revit that will help in minimizing the drafting or design process is developed with Revit application programming interface (API). By using Revit APIs building objects information from BIM model designed can be extracted by filtering that particular object with rest data like `FilteredElementCollector(_doc).OfClass(typeof(Wall)).Cast<Wall>().ToList();` in which

all the objects related to walls are filtered and can get the threshold values need to be checked with conditions satisfied. As discussed above, some of the information from BIM model can be easily obtained, some values must be obtained by using simple derived attributes, and some values need extended data structure to obtain those threshold values to check. Table 5 shows all the bylaws and building code rules that have been translated into computer interpretable format using C# language, where each rule is represented with building object, condition to be satisfied, attribute values to be derived from BIM model, threshold values to be checked for that object with operator, and the final column shows the formula/mathematical expression for those rules as represented in FOL equations and flow chart explained above. Table 6 in Appendix B shows all the building codes related to framing design from the Alberta Building Code 2014, which is represented based on building objects, conditions to be satisfied, and threshold value to be checked with respect to building object.

Table 4. Examples of Rule-based Knowledge for Checking Residential Building

Requirement Sources	Domain Knowledge					Formula/mathematical expression.
	Object	Condition	Attributes	Operator	Threshold	
Edmonton Zoning Municipal Bylaw	RF1 Zone Setback	Distance from building corner to Lot end.	Front setback distance	Minimum	6.0 m.	$d_{fs} \geq 6.0m$ \wedge
			Rear setback distance	Minimum	7.5 m.	$d_{rs} \geq 7.5m$ \wedge
			Side setback	Minimum	1.2 m.	$d_{ss} \geq 1.2m$
		If corner site, attached garage facing flanking public roadway.	Rear setback distance	Minimum	4.5 m.	$d_{rs} \geq 4.5m$
			Side setback	Minimum	20 % of site width	$0.2 \times w_s$ $\leq d_{ss}$ $\leq 4 \cdot 5m$
		Maximum		4.5 m.		
		If corner site, building facing front/side lot line and side setback abutting flanking side lot line If corner site, no attached garage and building faces flanking side of lot line	Side setback	Minimum	3.0 m.	$d_{ss} \geq 3m$
	Side setback		Minimum	4.5 m.	$d_{ss} \geq 4.5m$	
	Height		Maximum	10 m.	$h_{house} \leq 10m$	
	Single detached Housing	If attached garage faces flanking side lot line	Lot Area.	Minimum	250.8 m ² .	$A_{lot} \geq 250.8m^2$ \wedge
Lot Width.			Minimum	7.6 m.	$w_{lot} \geq 7.6m$ \wedge	
Lot Depth.			Minimum	30 m.		

	Duple x housing	All type of houses	House area.	Minimum	28 % of site.	$d_{lot} \geq 30m$ $A_{house} \geq 0.28 * A_{site}$
		Property/Lot lines				
		Site coverage, Site area > 300 m ²	Accessory Building area.	Minimum	12 % of site.	$A_{accessory} \geq 0.12 * A_{site}$
			Principal building with attached garage area.	Minimum	40 % of site.	$A_{principal} \geq 0.4 * A_{site}$
		Site coverage, Site area < 300 m ²	House area.	Minimum	28 % of site.	$A_{house} \geq 0.28 * A_{site}$
	Accessory Building area.		Minimum	14 % of site.	$A_{accessory} \geq 0.14 * A_{site}$	
	Principal building with attached garage area.		Minimum	42 % of site.	$A_{principal} \geq 0.42 * A_{site}$	
	Semi-Detached	Property/Lot lines	Lot Area	Minimum	300 m ² .	$A_{lot} \geq 300m^2$
			Lot Width	Minimum	10 m.	$w_{lot} \geq 10m$
			Lot depth	Minimum	30 m.	$d_{lot} \geq 30m$
		Site coverage	House area	Minimum	28 % of site.	$A_{house} \geq 0.28 * A_{site}$
Accessory building area			Minimum	12 % of site.	$A_{accessory} \geq 0.12 * A_{site}$	
Principal building with garage area			Minimum	40 % of site.	$A_{principal} \geq 0.40 * A_{site}$	
Property/Lot lines	Lot area	Minimum	488.4 m ² .	$A_{lot} \geq 488.4m^2$		
	Lot width	Minimum	14.8 m.	$w_{lot} \geq 14.8m$		
	Lot Depth	Minimum	30 m.	$d_{lot} \geq 30m$		

	Housing	If dwelling facing side lot line.	Lot width	Minimum	12 m.	$w_{lot} \geq 14.8m$ $d_{lot} \geq 30m$ $w_{lot} \geq 12m$
Alberta building code, volume 2: part 9.	Walls	Element Spacing	Studs	Maximum	24 inches.	$S_{studs} \leq 24 in.$
	Interior walls	Interior walls, building supports one floor where attic accessible by stairs	2X4 stud	Maximum	3.6 m. height and 400 mm. spacing.	$S_{studs.interior} \leq 16 in.$ \wedge $h_{studs.interior} \leq 3.6m$
			2x4 stud	Maximum	3.6 m height and 600 m. spacing.	$S_{studs.interior} \leq 24 in.$ \wedge $h_{studs.interior} \leq 3.6m$
	Exterior walls	Exterior wall, roof load with or without attic storage.	2x4 stud	Maximum	2.4 m height and 600 m. spacing.	$S_{studs.exterior} \leq 24 in.$ \wedge $h_{studs.exterior} \leq 2.4m$
			2x6 stud	Maximum	3.0 m height and 600 m. spacing.	$S_{studs.exterior} \leq 24 in.$ \wedge $h_{studs.exterior} \leq 3.0m$
	Openings	Exterior corners	Studs	Minimum	2 No.	$No_{studs.corner} \geq 2 no.$
			studs	Minimum	3 No.	$No_{studs.opening} \geq 3 no.$
studs			Minimum	2 No.	$No_{studs.opening} \geq 2 no.$	

	Top plates	Non-loadbearing walls	Top plate	Minimum	2 inches one stud with same width as wall studs.	$No_{toplates} \geq 1 no.$ \wedge
		Loadbearing walls	Top plate	Minimum	2 inches two studs with same width of studs.	$t_{toplate} = t_{wall.stud}$ $No_{toplates} \geq 2 no.$ \wedge
	Floor	Joist cantilevered distance beyond their support	2×8 joist	Maximum	16 inches	$t_{toplate} = t_{wall.stud}$ $d_{floor.cantilever} \leq 16 in.$
			2×10 joist	Maximum	24 inches	$d_{floor.cantilever} \leq 24 in.$
	Column	Column width		Minimum	Width of Supporting member.	$w_{column} \geq w_{support}$

3.3 Object-oriented representation

For a model to be checked automatically for building regulations, it is a primary requirement to have object-based building model with required level of information for effective checking. BIM objects created will have parametric data and properties such as their element name, size, location, finishes, and their abstract information like quantities, calculations and so on because this semantic enriched information BIM models will be used in digitizing the different stages of building lifecycle, including initial requirements, design, construction, maintenance, and operation. As such, EXPRESS Data Manager (EDM) model checker is an object data based system used in the HITOS pilot project in Norway, where building model data is stored and accessed through EDM model server in IFC format, and information for checking are parametrized, mapped with associated building objects and is executed using Solibri Model Server (Malsane et al., 2015).

BIM model with required level of information will make extracting the information needed more efficient, as models developed with more details makes it more convenient for developing automated code compliance checking. This study identifies the information model about framing and lot design boundaries for compliance checking. The required information for checking from model is retrieved from the specific objects it needs as those are mapped to the compliance checking rules. The excerpt of framing information in this prototype is shown in Figure 3.12. “Studs” and “plates” are the classes created within visual studio to represent the BIM information required for checking related to studs and plates. Modelling details in Autodesk Revit related to these objects are mapped to those classes to facilitate compliance checking. As shown in Figure 3.12 “BuildingComponent” is base class that has general information about building elements, and “wall”, “Plates”, and “Studs” are inherited from “BuildingComponent”. “Framing” is inherited from interface “Project” which has the properties related to building information and is associated with “studs” and “Plate”. The

function of “GetSpacing”, “GetMaxHeight”, “GetCornerStudDetails” and “GetOpeningsStuds” is attached to “studs”, which utilizes the information related to framing to extract maximum spacing between the studs, maximum height of studs, details related to studs present in corner and at connections, and details related to studs placed around openings respectively for automated code compliance.

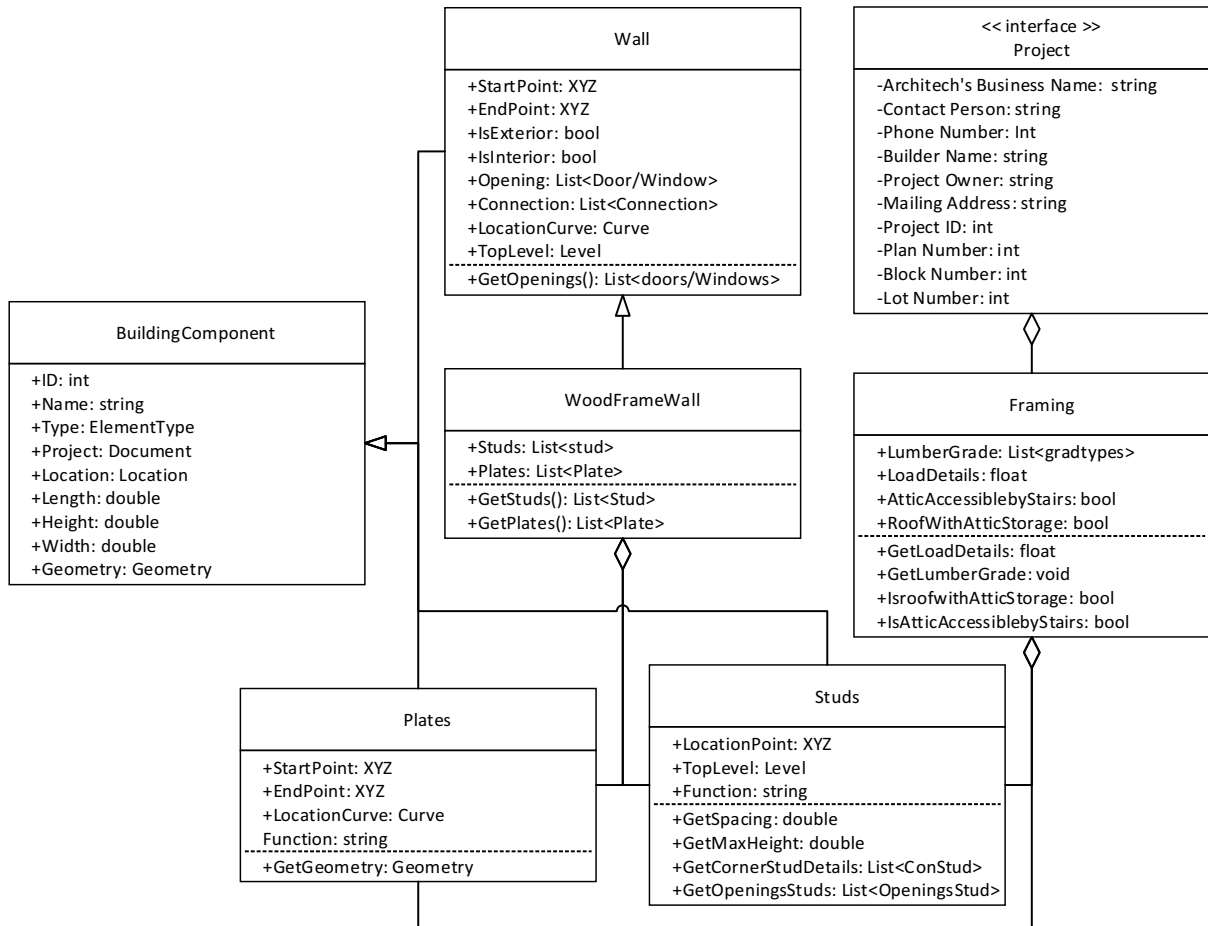


Figure 3. 12. Excerpt of framing information for checking using UML.

CHAPTER 4. CASE STUDY

In this chapter, a case study is presented for validation and verification of the proposed methodology for automated zoning and framing code compliance for light frame residential building in Edmonton. The validation of the developed add-on software application is carried out from inception and to completion, including the testing of each command for its intended purpose. A residential building with attached garage is taken as an example, for which the bylaws and building framing code rules have been mentioned in Table 5. Rule-based knowledge checking for residential buildings have been implemented and validated in this prototype called DCheck. DCheck is an add-on application for Autodesk Revit that performs the automated code checking of the zoning regulation of Edmonton and the sections of the building code related to framing of residential buildings. This add-on software application can be used for approving the BIM models for construction and by the designers to make sure the design is correct. Regarding this study, three major topics are addressed in this case study related to the developed add-on software application: (1) Edmonton zoning bylaws implementation; (2) Alberta Building code 2014 regulations related to framing for light framing residential buildings; and (3) DCheck extended data structure for extracting the information from BIM model.

A residential building design has to be modelled in Autodesk Revit as required, with a certain level of details for more efficient automated-checking process. Before starting checking, users are required to give some information related to the project, which is same information house owners or contractors used to provide while submitting 2D CAD drawings for approval. Figure 4.1 and 4.2 show the 3D model of residential buildings with attached garage modelled in Revit for the purposes of this case study: the brown coloured area represents the lot dimensions for this house.

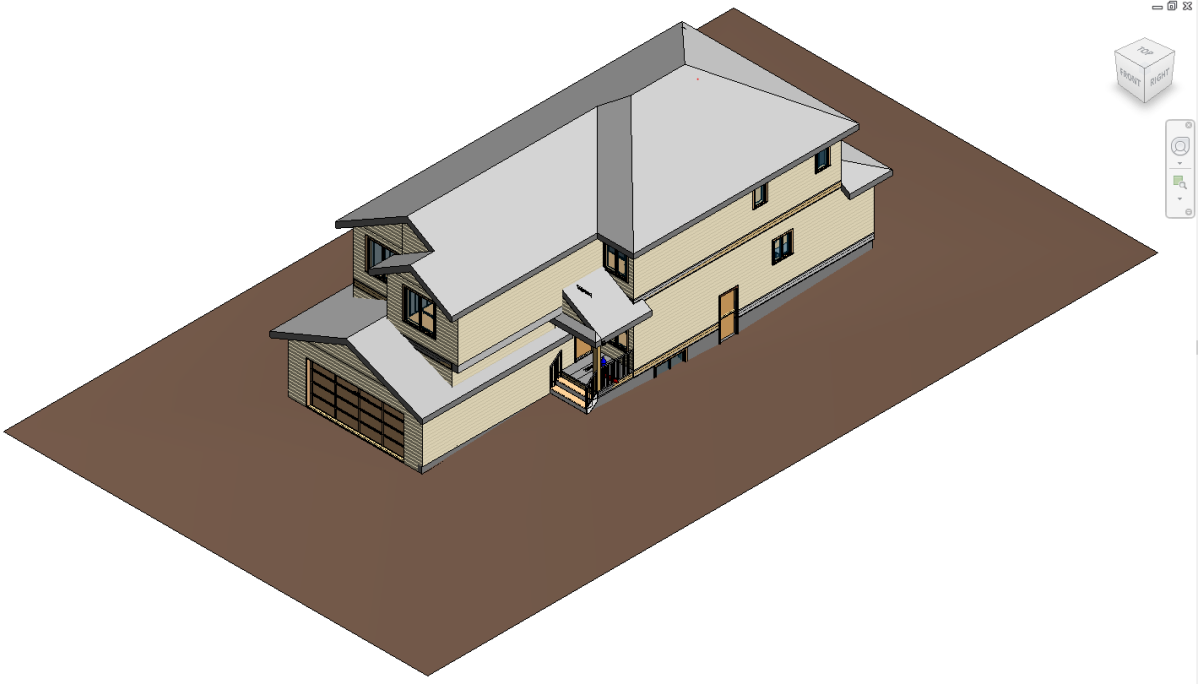


Figure 4. 1. 3D Model of Single detached house with Attached Garage (case study model 1).

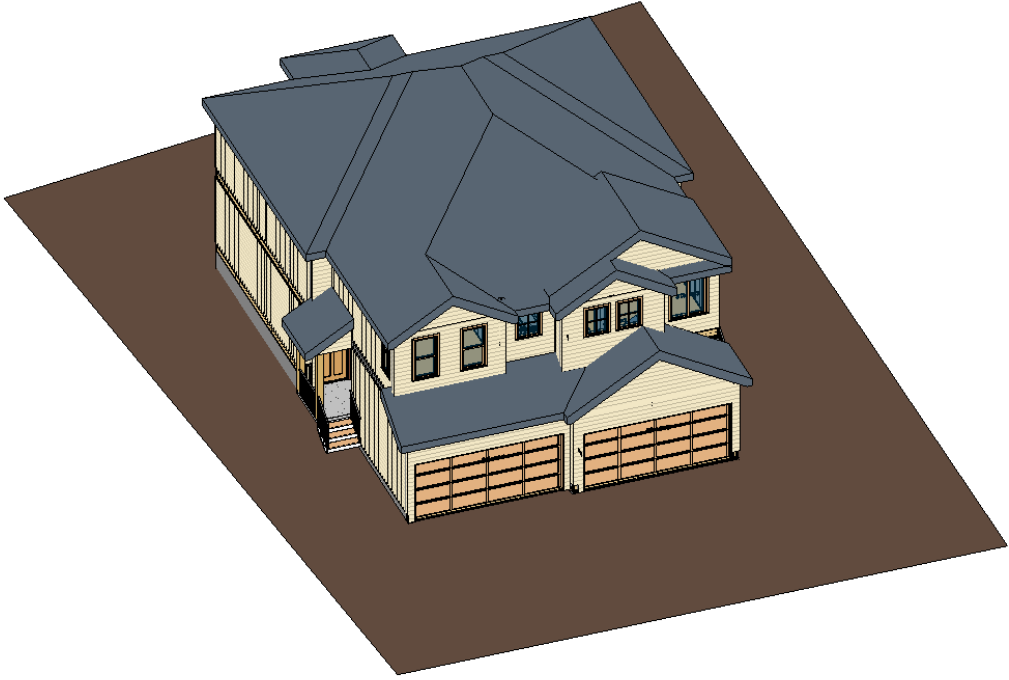


Figure 4. 2. 3D Model of semi-detached house with Attached Garage (case study model 2).

Figure 4.3 and 4.4 are the floor plan views of single detached house model and top view of semi-detached house plans respectively. The major part of the rule checking process is the rule definitions. Rules apply to each part of subset of BIM model data. Building model preparation and the development of model views with appropriate subset of data required for rule checking is important. Model views are a standard form of representation of BIM data that allows extraction of meaningful required data from the model. It typically includes geometric, topological, and functional properties of a respected building object. To extract all the required information from the BIM model for compliance checking, first the building model has to be developed with required level of details (LoD). The LoD is an important factor to be considered during development of each objects, as higher the LoD is the more details within the BIM model. For this study, development of BIM model with LoD of in between 300 to 350 will be good for getting required and complete data. A model developed with LoD in that range will be detailed for every specific system, object or assembly in terms of quantity, size, shape, location and orientation with non-graphical information and adds requirements on interfaces with other object of building.

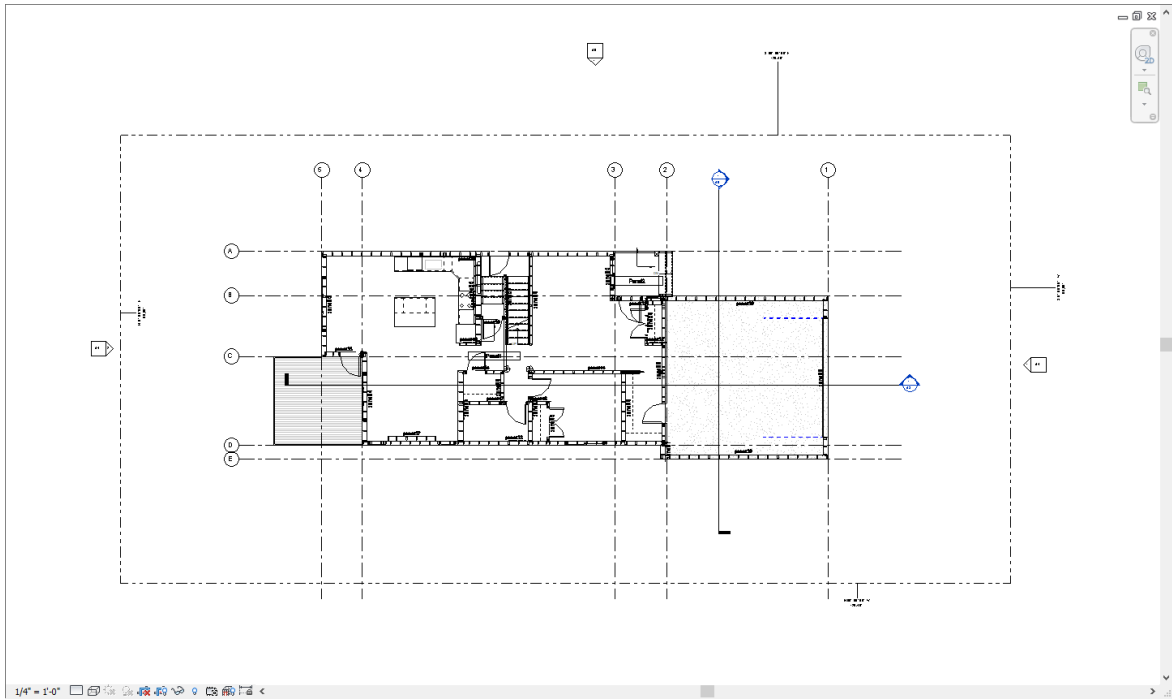


Figure 4. 3. Floor Plan of single detached House with Attached Garage (case study model 1).

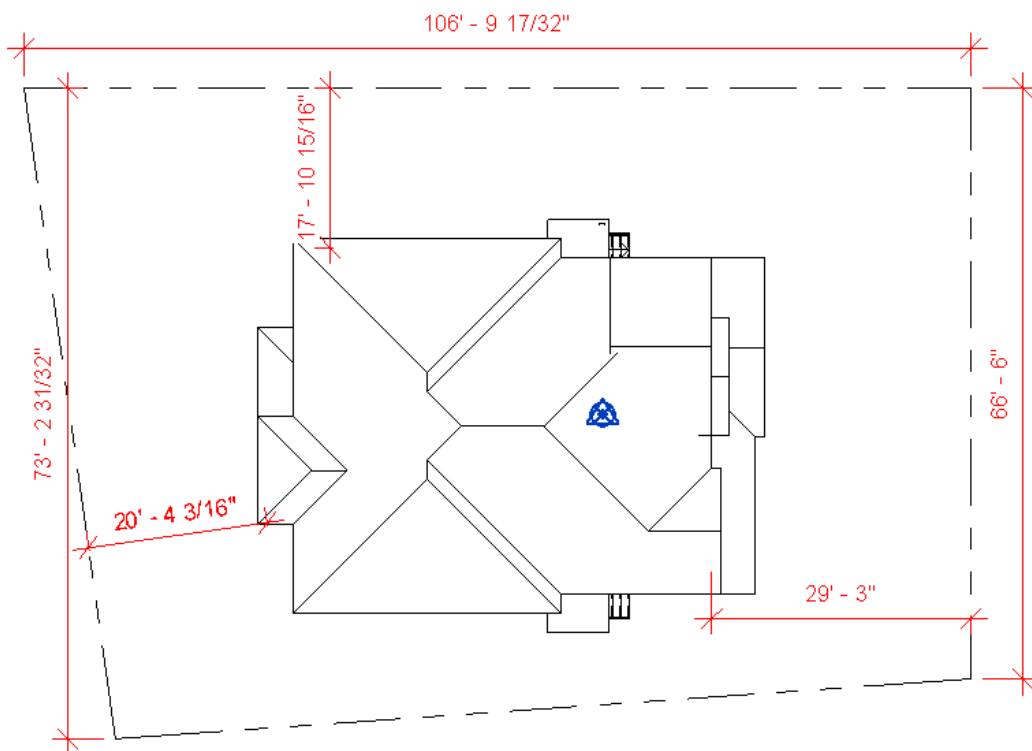


Figure 4. 4. Top view of semi-detached House with Attached Garage (case study model 2).

The other important part of rule checking process is rule definition. The rules are written in

human-oriented languages that require significant domain knowledge to translate it into computer interpretable format. There are many ways of doing this part: in this study we have used C# language for interpreting natural language-based regulations into computer interpretable format. In order to interpret the bylaws and building code, the sentences of the bylaws and code were analyzed to identify the threshold value of rules to be satisfied related to building objects. Rules are then represented based on object with conditions to satisfy, attributes related to that objects and threshold value to be checked with operator. This method of representation is highly efficient to make components pool of building related objects.

As mentioned in the methodology above, automated rule-based checking for design checking consists of four major steps (1) Rules Translation, from natural language to computer interpretable format, (2) BIM model preparation, development of 3D model with certain level of details and development of building model views for extracting information from BIM data model to check for rules that have been translated in first step, (3) Rule Checking, where building model is checked against the building code under DCheck platform, and (4) checking reporting, which is the last step and gives the end results of compliance in text format and highlights the objects associated with failed rules.

4.1 Bylaw Checking

As if the user selects the zoning type as single detached residential zone (RF1), then the RF1 type of zone is being considered as an object here and the check for height of the building as per bylaw, threshold value of 10 m, with operator sign less than or equal too. That is, the building height cannot exceed 10 m for RF1 type zone. Getting the total height of the building from BIM data depends on the type of roof the model has, as shown in Figure 3.6. For flat and sloped roof, the total height of building will be second floor wall height from ground level plus height of roof. If the roof is gable or hip or mansard or gambrel, then total height of building will be second floor wall height from ground level plus half of the roof height. These conditions

have been defined in DCheck extended platform to check if the roof is flat or sloped with accessing the co-ordinates of roof by using Revit API's for knowing geometrical property of roof type under "RoofType" class. Then, the user selects any of the three different house types that are permitted to be built in RF1 zone, and in this case the user selects single detached housing for which lot area should be greater than 250.8 m², lot width should be greater than 7.6 m, and lot depth should be greater than 30 m according to bylaw. If any of these regulations are not satisfied, then notification regarding the violation of rules and suggestions for correcting it have been encoded in the platform.

Depending on housing type in RF1 zone. If the house type is single detached house with site area less than 300 m² or semi-detached housing with site area less than 600 m² and the house is a principal dwelling with separate garage/accessory building, then the house can cover a maximum of 28% of the site/lot area and accessory building can cover maximum of 14% of the site area.

If the building has an attached garage and the site area is less than 300 m², then the building can cover a maximum of 42% of site area. If, however, the house type is single detached house with site area greater than 300 m² or duplex house or semi-detached house with site area greater than 600 m² or if house has a separate accessory/garage building, then principal dwelling can cover maximum of 28% of site area, and accessory/garage building can cover 12% of site area. If the building has an attached garage and the site is greater than 300 m² then building can cover maximum of 40% of site area. If any of these cases fail to satisfy, error message relating to failed criteria will be displayed with suggestion message to correct.

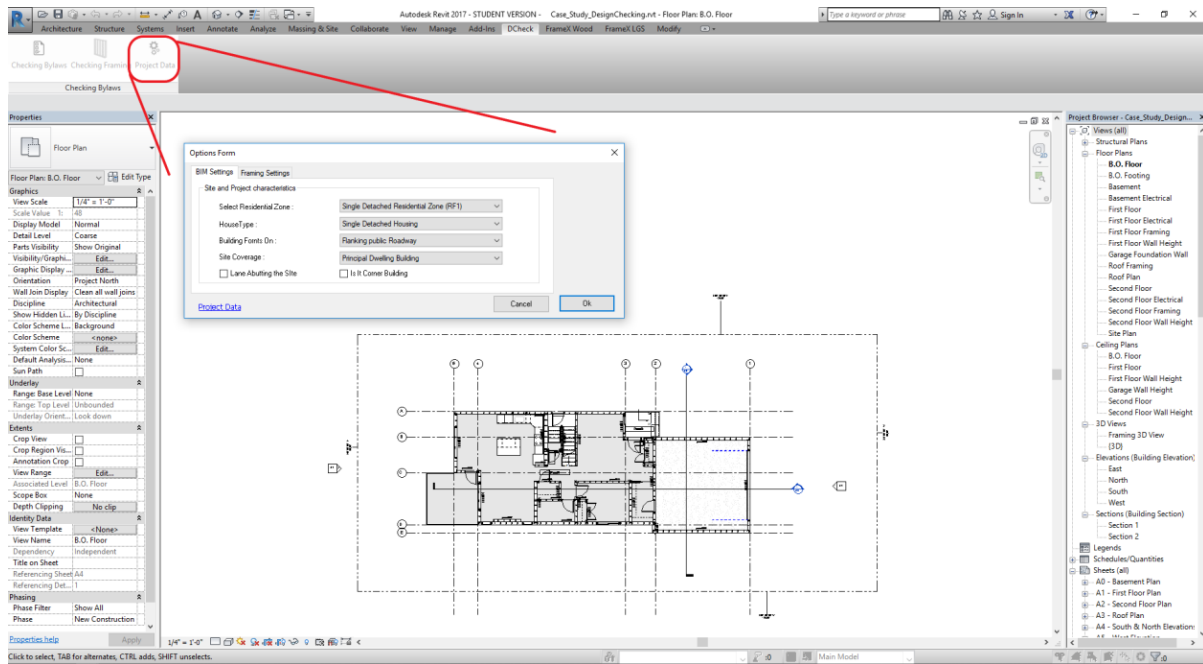


Figure 4. 5. Main User Interface of DCheck add-on.

DCheck has been developed as an add-on software application for Revit that performs the automated design checking. Figure 4.5 shows the main user interface of DCheck application; as shown, it consists of three main buttons on the Revit ribbon interface. “Checking bylaws”, “Checking framing” and “Project data”. Clicking on project data button user interface will appear where user can provide information related to zoning. There are two tabs. The first tab is for inputs related to bylaws where user needs to input some information, for example, the type of residential zone the model will be built from: Single Detached Residential Zone (RF1), Residential small Lot Zone (RSL), Low Density Infill Zone (RF2), Planned Lot Residential Zone (RPL), Small Scale Infill Development Zone (RF3), Semi-detached Residential Zone (RF4), Residential Mixed Dwelling Zone (RMD), Row Housing Zone, Urban Character Row Housing Zone (UCRH), Medium Density Multiple Family Zone (RF6). There is also an input for house type: single detached housing, semi-detached housing, duplex housing, limited group homes, garden suite, secondary suites, or minor home-based business. Site characteristics input is for information about whether it is a corner building, or whether the building fronts on to

flanking public roadway or front yard, or whether there is a lane abutting the site.

The image shows a software dialog box titled "Options Form" with a close button (X) in the top right corner. It has two tabs: "BIM Settings" and "Framing Settings". The "BIM Settings" tab is active and contains a section titled "Site and Project characteristics" with the following fields:

- Select Residential Zone : Single Detached Residential Zone (RF1) (dropdown)
- House Type : Single Detached Housing (dropdown)
- Building Forms On : Flanking public Roadway (dropdown)
- Site Coverage : Principal Dwelling Building (dropdown)
- Lane Abutting the Site
- Is It Corner Building

Below this is a section titled "Project Data" with a blue link. It contains two columns of input fields:

- Architect's Business Name : [text box]
- Contact Person : [text box]
- Phone Number : [text box]
- Fax : [text box]
- Builder Name : [text box]
- Project Owner : [text box]
- Mailing Address : [text box]
- Project ID : [text box]
- Plan # : [text box]
- Block # : [text box]
- Lot # : [text box]

At the bottom right are "Cancel" and "Ok" buttons.

Figure 4. 6. User Interface for Entering Project Details.

There is also an input for building characteristic information about whether building has attached garage or not. Figure 4.6 shows graphical user interface form developed for all the information related to the project that needs to be provided for approval: these are all the supporting information that needs to be provided while applying for approval for construction. Details about the architect's business name, contact person, phone number, fax address, builder name, project owner, mailing address and project related details like project ID, plan number, block number, lot number, i.e., identity number provided by the city for site where the proposed building will be built. By clicking on the OK button after entering all the details related to zoning and project related details, checking of model will run in background and by clicking on checking bylaws button, the user interface report form will be displayed as shown in Figure 4.7 with the report related to compliance checking. It will display the report about rules that failed with attribute name, reason for the failure and details about that rules, so that user can

easily understands the error in designs related to zoning bylaws. Figure 4.7 shows the zoning checking report with message for single detached residential house and Figure 4.8 shows zoning checking report for semi-detached residential house:

Start to checking

=====

Failed SETBACK :

- Front SETBACK (Front) = 5.09 m, it must not be less than 6 m.
- Rear SETBACK (Rear) = 5.62 m, it must not be less than 7.5m in the normal case.
- Side SETBACK (Left) = 2.45 m, it must not be less than 20% of the side width of the site.

=====

Finish !

Checking Failed.

So, from the text report like shown above, the user will be able to know the failed rules, reason for failure and can correct them as suggested.

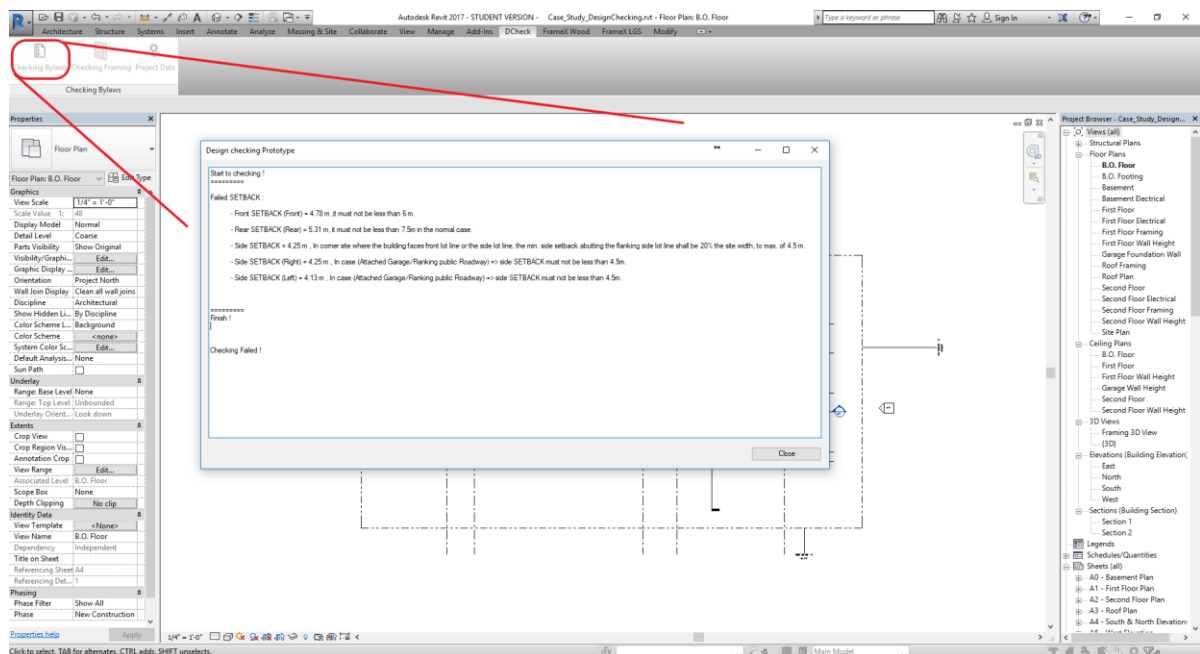


Figure 4. 7. Bylaw Checking Report for single detached house (case study model 1)

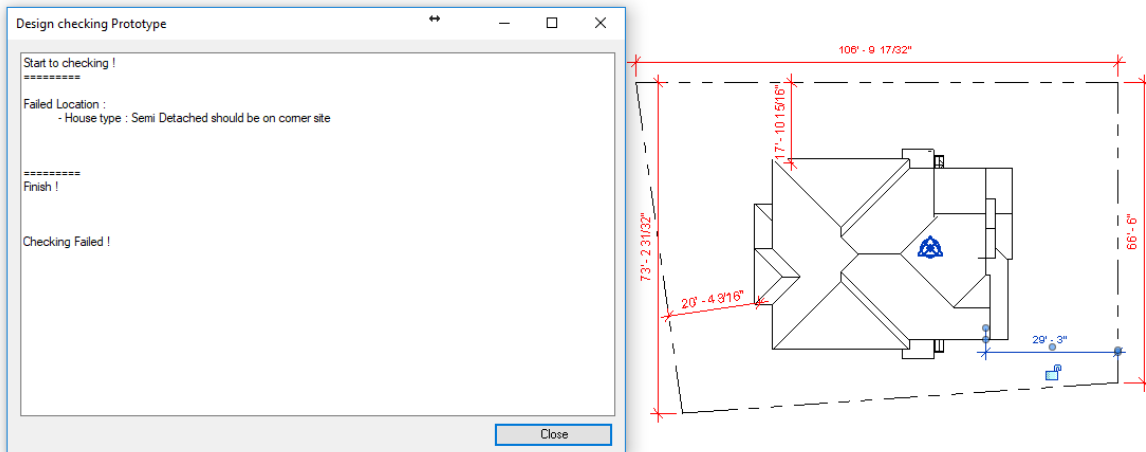


Figure 4. 8. Bylaw Checking Report for semi-detached house (case study model 2)

Once those errors have been corrected, then by again just clicking on checking bylaws button, it will run the checking again and display the same user interface report form with message whether all the rules have been checked successfully or failed with message related to failed rules.

4.2 Framing Checking

Another aspect that has been covered in this study is framing checking with Alberta Building Code 2014 for wood-frame houses built with different designs and specifications to provide safety, maximum occupational health, comfort, durable and cost efficiency. Wood framing consists of main structural members (i.e., framing) and sheathing. Many types of wood components can be used in wood-frame construction. Lumber is generally referred to by nominal dimensions, which are larger than actual dimensions. Considering the wood framing design guide and advanced framing techniques referred to as optimum value engineering framing that will reduce wood material use in construction in structurally unnecessary places, all those rules have been included for checking to get optimized design of framing. Figure 4.9 shows the framing of walls and floors of the case study building; the details of framing are

accessed by their naming description as specified in model. Framing of this model has been done using an add-on called FrameX, in which all the wall studs are named as “Vertical” with panel number, studs placed at corners and ends are as “End” and “Con”, bottom and top plates as “BTrack” and “TTrack”, and so on. Details related to each lumber have been extracted by using naming details specified by the FrameX add-on application.

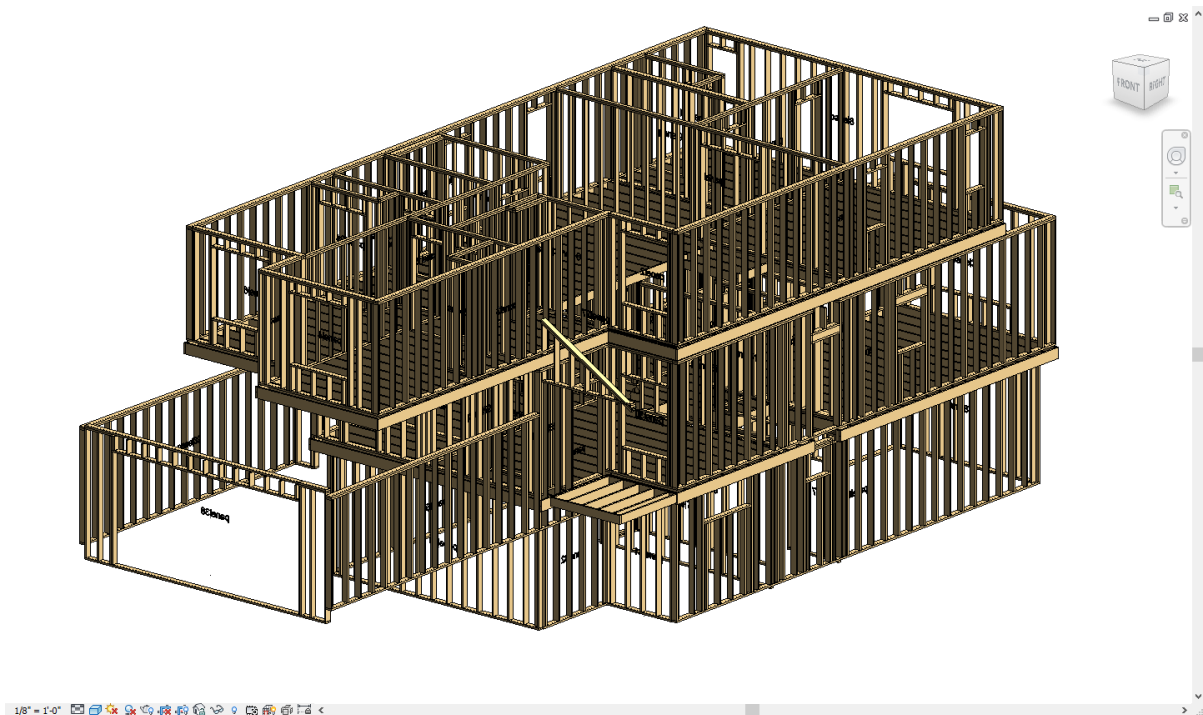


Figure 4. 9. Framing of Walls and Floors of House.

The framing guidance rules are collected from the Alberta Building Code 2014 Div B Part 9, Advanced framing construction guide by APA, the Engineered Wood Association, Canadian wood-frame house construction by Canada Mortgage and Housing Corporation, and Design of wood framing by residential structural design guide.

Except in some special cases, the framing of a residential building has to be designed as per authorized designing guidelines, excluding those exceptional cases, and all the remaining guidelines for checking the framing design have been collected in Table 6 in Appendix B. The framing rules that have been integrated in the prototype add-on software application for

checking considering the load details and type of wall, whether it is interior or exterior, and dimensions of studs, and checking for spacing and height of studs depending on wall support load conditions. If the wall is exterior with 2×4 stud dimension and if the walls are supporting roof load with or without attic storage plus two floors, then stud spacing should be less than 12 inches; or if walls support roof load with or without attic storage plus one floor, then spacing should be less than 16 inches; or if walls support only roof load with or without attic storage, then spacing should be less than 24 inches, and also, for every condition the maximum height of studs should be less than 3 m. The same kind of rules apply to the interior walls too as shown in the flow chart below. If any of the checking has failed, then suggestion message for failed rules are displayed.

The user needs to provide some of the information related to framing before starting to check through the user interface by clicking on project data button and entering details required about framing in the second tab of interface as shown in Figure 4.10. This second tab is where the user needs to select lumber grade and whether it is structural or No. 1 and No.2 or No.3 or construction or standard grade lumber. The user provides live load and dead load details. Information about whether the attic is accessible by stairway or not, and whether the roof is with or without attic storage. All the framing rules that have been integrated in this prototype add-on software application are mentioned in Table 5 and are represented in the same way as bylaws with building object, attributes, condition and threshold value with operator.

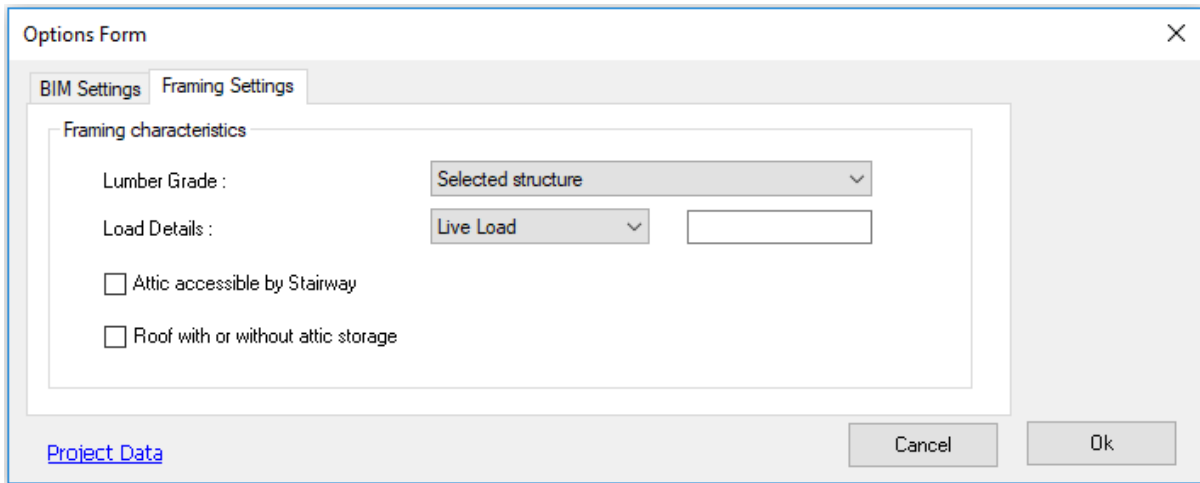


Figure 4. 10. User Interface for Entering Framing Details.

After entering all the details required for this check, then by clicking on framing checking button on ribbon, a checking window will pop up with the checking report related to framing as shown in Figure 4.11 and 4.12 The failed rules related to framing design for single detached and semi-detached house are as shown below:

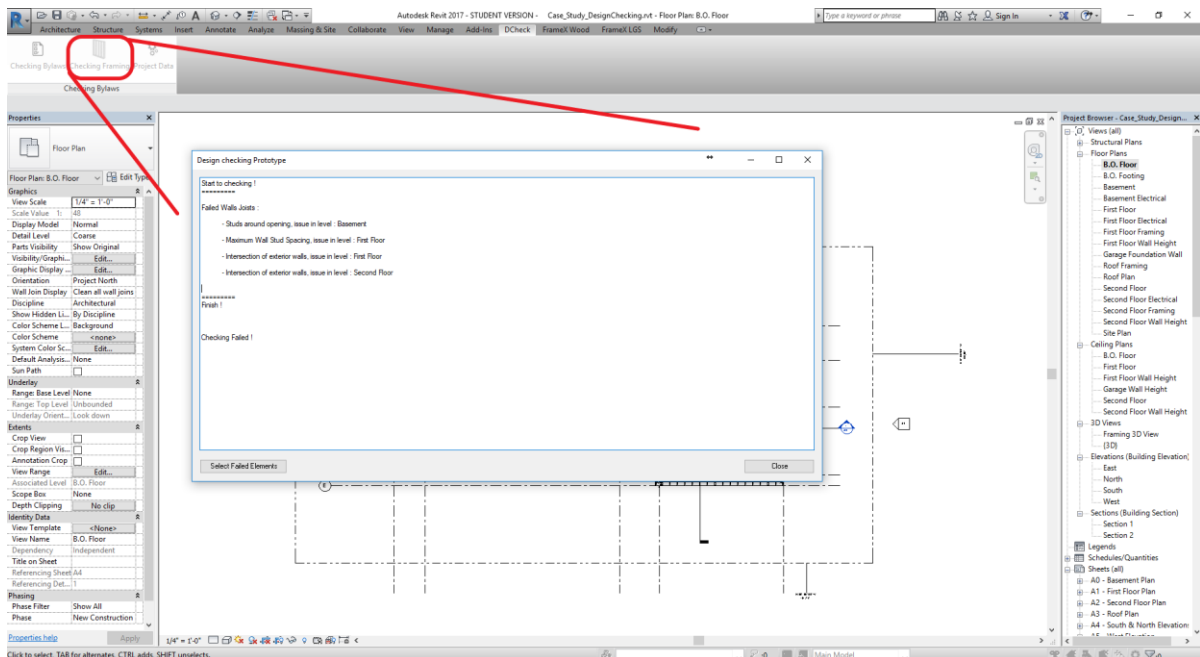


Figure 4. 11. Framing Checking Report for single detached house (case study model 1).

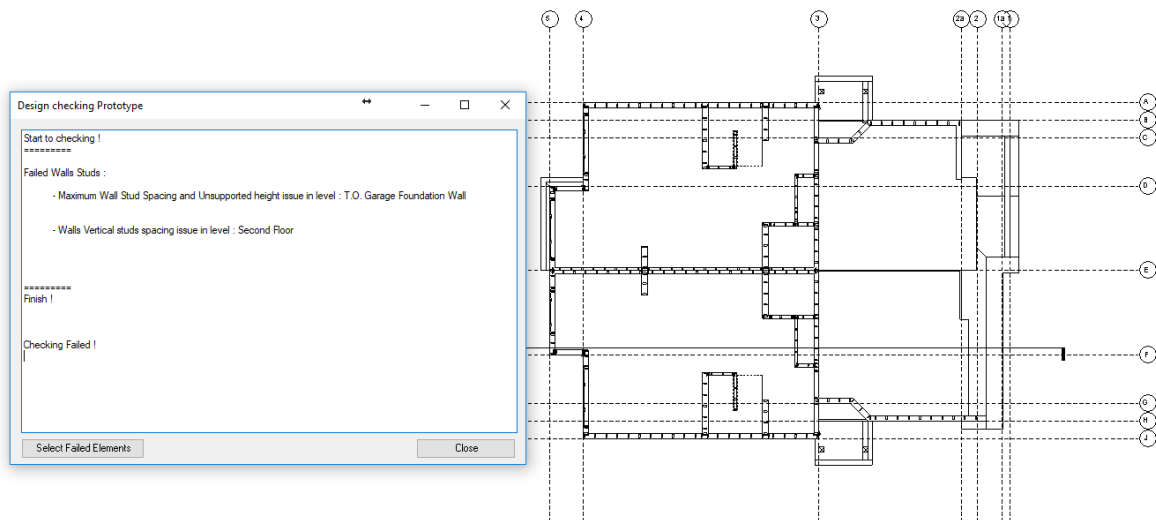


Figure 4. 12. Framing Checking Report for semi-detached house (case study model 2).

Start to check

=====

Failed Wall Studs :

- Studs around opening, issue in level: Basement.
- Maximum wall stud spacing, issue in level: First Floor.
- Intersection of exterior walls, issue in level: First Floor.
- Intersection of exterior walls, issue in level: Second Floor.

=====

Finish !

Checking Failed.

With failure report in text format as shown above, the user will be able to know the failed rules, reason for failure and can correct them as suggested.

By clicking on select failed elements, it will select the building objects that are related to failed rules so that the user can easily identify and make corrections as shown in Figure 4.13, where it has selected the door opening and studs beside the door opening because the opening is more

than 3 m wide. In this case, the studs beside opening should be tripled as per building code, i.e., “In lintel opening of wall if it is greater than 3 m long then studs should be tripled on each side of the opening, with 2 from bottom of lintel to top of bottom of wall plate, and 1 from bottom of top wall plate to bottom wall plate.”

In the case study model we have 192 in (4.87 m) of opening and have only two studs placed around the opening on each side, one king and one jack. But as per the above rule, if the opening is greater than 3 m (118.1 in), it should have three studs on each side of opening, and because of this, it is highlighted in the basement floor plan as the error message displayed.

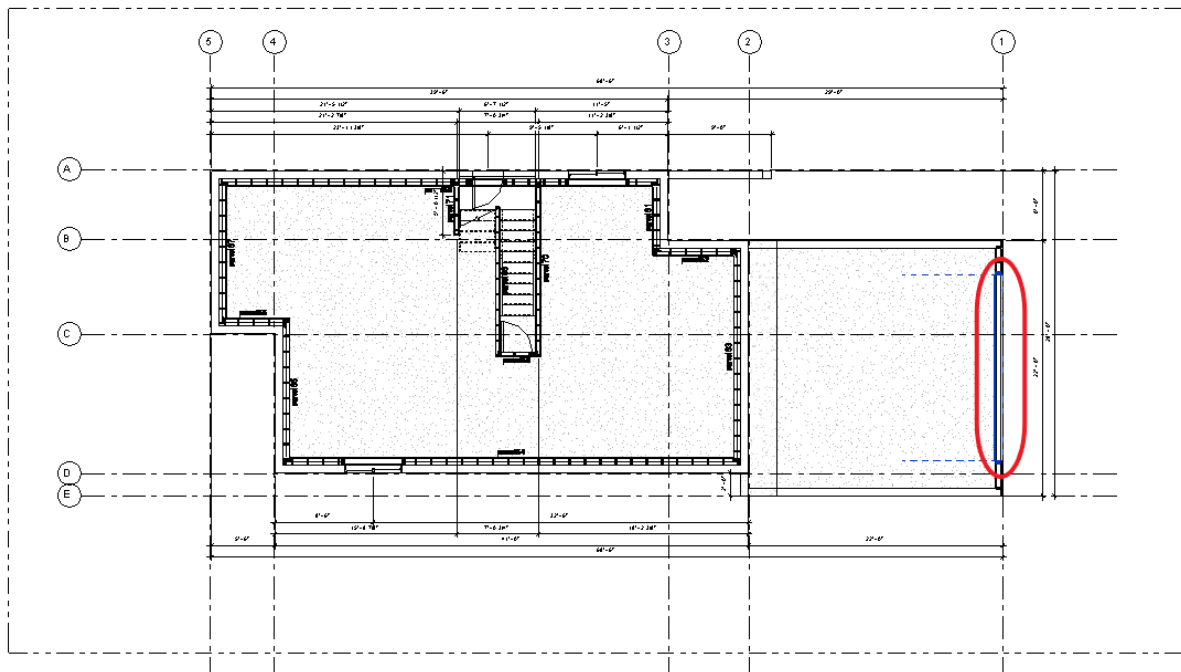


Figure 4. 13. Highlighting Failed Rules Related Objects in Basement Floor.

After correcting all the errors and by running the checking process again, if there are no errors then it will show the text report saying checking successful as in Figure 4.14 below, which indicates the model is good for the approval of construction as designed.

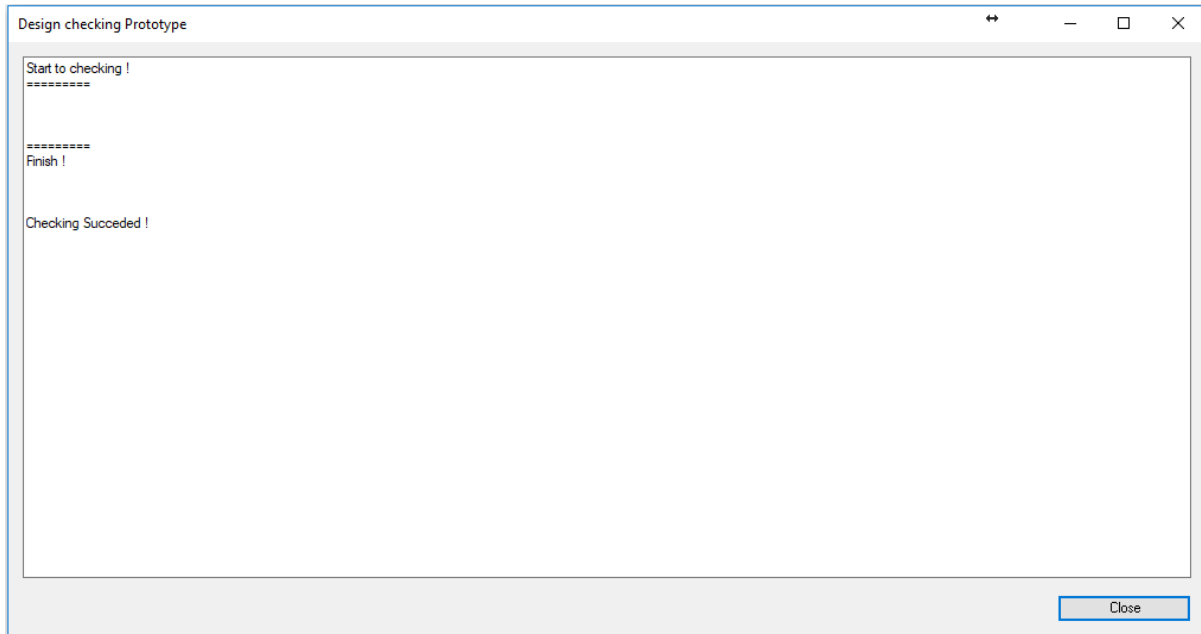


Figure 4. 14. Final Check Results After correcting all the Errors.

4.3 Discussion

Exhaustive studies have been made in this field of research by many researchers around the world, where it all started by the logical organization of rules and regulations, followed by the structuring of regulations in decision tables. It is because of these initial research efforts and the efforts of those researchers who continued their research in this field that progress has made in this automated code checking field with various different approaches and technologies. However, to this day, no one in the AEC industry has been utilizing or benefiting completely from this technology. The Singapore CORENET project was an early initiative started by a government organization about 23 years ago for the checking of 2D building plans submitted online for code compliance. And then around 1999, the Solibri company developed an application called Solibri model-checker (SMC) for checking 2D plans. And even after over two decades since the start of the CORENET project, there has not been much progress in the development of an automated checking process, and Solibri is the only commercial software available for checking some aspects of building design like clash detection, and space validation, while also providing quantity take-off capabilities. So, the applications developed

should provide easy way for future updated and should be user friendly, with development of application based on building objects as represented in domain knowledge table above provides easy way of changing or updating the threshold values in according to update of building codes. With the use of BIM technology by AEC professionals, automated code checking has gained more interest. Even with most of the AEC industry using BIM models for the complete process of construction, the process of getting approval from municipalities for construction work has remained very much a 2D process, which seems like a waste of resources and efforts for companies to use BIM modelling software for design and construction process from this prospective. Effectively, the only progress has been the submitting of electronic drawings to authorities for approval.

With so many constraints in the process of automating the building code compliance checking, it is good to see the progress in this field. The major problems faced related to automation is the translation of building code from natural human readable language to computer interpretable format. Finding the best and most suitable way of translating all the building regulations for fully automated checking process is very important, with representation of the building code based on objects, attributes, conditions and threshold value with operator, as was done in this study, as a way to help better understand the building code and bylaws for efficient interpretation. Another problem is the BIM model preparation, i.e., the level of details the BIM model should have for effective checking and extraction of details from BIM model. The development of extended data structure platform results in a better and more effective way of automated code compliance checking because it enables all building objects to achieve a certain required level of detail.

Many researchers around the world, in both commercial and academic fields, are working in this field to develop better and more efficient ways of automating the building code checking process. The ultimate benefit of automated code compliance is obtained when the government

authorities make BIM models mandatory for all the construction approval processes and automated checking application are used by both designers and government regulatory agencies responsible for the reviewing of designs. If used by both designers and government authorities, the amount of time required for the reviewing process will be greatly reduced and the efficiency in checking will be increased with less efforts and man power.

CHAPTER 5. CONCLUSION AND FUTURE RESEARCH

In this chapter, conclusions, key contributions, and recommendations for future research are presented. First, the summary of the automated code compliance checking add-on software application developed is discussed briefly. Then, contribution of this research is summarized; finally, recommendations for future study are proposed.

5.1 Summary

This thesis presents the automated checking of framing design and zoning regulations according to the city of Edmonton municipal bylaws for light frame residential buildings modelled in Autodesk Revit. In this study, the translation of natural language building code into computer interpretable format is done using C# language. The representation of the building code based on building objects will make it easier to understand the regulations and will help to translate the regulations into computer-readable format more accurately, which then makes the job of automation of code compliance easier when it is time to develop the add-on software tool for a different province or jurisdiction where the building codes are different. If required, only the threshold values will be changed, which can be easily accomplished as per the explanation in the methodology section. Because of this, DCheck extended data structure platform can be used for other provinces too, just by changing values according to respective provinces and this application will perform checking accordingly. Also, with the classification of building regulations supports the development of the extended data structure platform. Classification of rules according to complexity in interpreting building regulations into computer interpretable format and based on retrieving the information from BIM models helps to translate rules completely based on classification. The first type of classification of rules includes data that can be easily accessed from BIM model. The second type requires some expert knowledge where the required information should be derived from BIM model. The

third type of rules are the ones that need to be simplified and analyzed, and the information requires an extended data structure. Extraction of information from BIM model to check with the threshold values of code is a major challenge. All the information in BIM model is not explicitly available, even with models developed with 300 to 350 LoD, some information has to be derived from BIM model introducing some new variables relating to building objects in BIM model.

The DCheck add-on application for Revit has included the rules listed in Table 5 related to zoning checking for City of Edmonton municipal bylaws and wood framing design checking according to the Alberta Building Code 2014. The add-on software application can be used at any point during the designing for checking the model, so that the designer can correct any errors if present during the design progress. With the use of automated code compliance application by both designers and government approval authorities, the approval process can be made with less effort, fewer errors, and in less time.

5.2 Research Contributions

The research presented in this thesis offers the following contributions:

- (1) Object-based representation of building code and bylaws. Building rules are represented in accordance with respective building object with their attributes, conditions, and threshold valued with operator to be satisfied. This kind of knowledge formularization makes it easy to understand the regulation and know the required threshold value to be checked for that building object in the model with specified conditions.
- (2) Classification of building rules into three categories, from easy to difficult level, based on complexity for interpretation into computer readable format, and based on the extraction of information from BIM model. The classification of building rules makes it very convenient for developing the checking add-on software application platform.

The checking system can be developed in stages, which gives the user a clear understanding of checking process with rules that are incorporated in application system.

- (3) Automated checking process of zoning design in accordance with Edmonton zoning bylaws. The add-on application for Revit, DCheck, with user interface developed will check for the design regulations of zoning with minimum user interaction and gives results in textual format, indicating the failed rules and suggestions to correct.
- (4) Automated checking process of framing design in accordance with Alberta Building Code 2014. Apart from some special or critical cases, framing of residential building have been done in accordance with the building code, with automated checking of building code for framing where designers can check for any errors effortlessly and in no time. The checking report is provided in textual format and the building objects related to errors are highlighted, which then can be corrected.
- (5) Simplified DCheck platform for extracting the information from BIM model for code compliance. This platform can be used to check for different jurisdictions too by changing the threshold value with respect to different municipal zoning bylaws.

5.3 Limitations and Recommendation for Future Work

5.3.1 Research Limitations

This research is subjected to a few limitations as follows:

The building regulations have been hardcoded into the add-on software application system by using the C# coding language, which makes it difficult for the user if there is a requirement to change anything or for updating the regulations if the user does not have any basic knowledge related to coding languages. And every time the building codes will be updated, the checking platform has to be changed as per updates with regard to new codes.

This study covers only few aspects of the checking for building permit approval. Automated checking for structural design analysis of the wood framing is not considered in this study, because of which in some special cases structural design as to be checked manually.

The checking application is built as an add-on software application for Autodesk Revit, so the models developed in other modelling software will not be able to be checked with this application, and this add-on is built on Revit API platform so except for Revit files, no other file formats can be checked.

5.3.2 Future Research and Improvements

Based on the research presented here, the following would be the recommendations for future work:

Extension of automated checking process with other aspects of the building codes and bylaws for developing a complete automated building rules checking process incorporating all the rules and regulations for building permit approval.

Make the add-on software application more user-friendly by creating an additional user interface for all the threshold values so the user can easily update or change values if required. This will also help in using this application for other jurisdictions too just by changing those values according to their municipal regulations.

Development of mathematical representation of regulations based on the building variables for better understanding and translation of natural human readable language into computer interpretable format.

Create better way of communication in connection with same add-on software application system between the architect or contractor or owner applying for building permit and authorities responsible for proving approval for better understanding of errors needed to be corrected.

REFERENCES

- AEC3 website. (2012). Retrieved from http://www.aec3.com/en/5/5_013_ICC.htm
- Autodesk Inc., website. (2018). Retrieved from <https://www.autodesk.com/>
- Automated compliance checking using building information models. (2010), (September), 2–3.
- Chen, G., & Luo, Y. (2017). A BIM and ontology-based intelligent application framework. *Proceedings of 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference, IMCEC 2016*, 494–497. <https://doi.org/10.1109/IMCEC.2016.7867261>
- Choi, J., Choi, J., & Kim, I. (2014). Development of BIM-based evacuation regulation checking system for high-rise and complex buildings. *Automation in Construction*, 46, 38–49. <https://doi.org/10.1016/j.autcon.2013.12.005>
- Clayton, M. J., Fudge, P., & Thompson, J. (2013). Automated Plan Review for Building Code Compliance Using BIM. *Proceedings of the 20th International Workshop, 1-3 July*, (July 2013), 1–10. Retrieved from www.researchgate.net/publication/254862600
- Dimyadi, J., & Amor, R. (2013). Automated Building Code Compliance Checking – Where is it at? *Proceedings of the 19th World Building Congress: Construction and Society, 5-9 May*, 172–185. <https://doi.org/10.13140/2.1.4920.4161>
- Dimyadi, J., Solihin, W., & Hjelseth, E. (2016). Classification of BIM-based Model checking concepts. *Journal of Information Technology in Construction*, 21(November), 354–370.
- Ding, L., Drogemuller, R., Rosenman, M., & Marchant, D. (2006). Automating code checking for building designs - DesignCheck. *Cooperative Research Centre (CRC) for Construction Innovation*, 1–16. <https://doi.org/http://ro.uow.edu.au/engpapers/4842/>
- Dym, C. L., Henchey, R. P., Delis, E. A., & Gonick, S. (1988). A knowledge-based system for automated architectural code checking. *Computer-Aided Design*, 20(3), 137–145.

[https://doi.org/10.1016/0010-4485\(88\)90021-8](https://doi.org/10.1016/0010-4485(88)90021-8)

Eastman, C. (2009). *What is BIM?* Retrieved from <http://bim.arch.gatech.edu/?id=402>

Eastman, C., Lee, J. min, Jeong, Y. suk, & Lee, J. kook. (2009a). Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), 1011–1033. <https://doi.org/10.1016/j.autcon.2009.07.002>

Eastman, C., Lee, J. min, Jeong, Y. suk, & Lee, J. kook. (2009b). Automatic rule-based checking of building designs. *Automation in Construction*, 18(8), 1011–1033. <https://doi.org/10.1016/j.autcon.2009.07.002>

Ebrahimipour, V., & Yacout, S. (2015). Ontology modeling in physical asset integrity management. *Ontology Modeling in Physical Asset Integrity Management*, 1–264. <https://doi.org/10.1007/978-3-319-15326-1>

Fenves, S., Garrett, J., & Kiliccote, H. (1995). Computer representations of design standards and building codes: US perspective. *International Journal of Construction Information Technology*, 3(1), 13–34. Retrieved from <http://fire.nist.gov/bfrlpubs/build95/PDF/b95012.pdf>

Global Reporting Initiative. (2014). Construction and Real Estate. Retrieved from <https://www.globalreporting.org/reporting/sector-guidance/sector-guidance/construction-and-real-estate/Pages/default.aspx>

Han, C. S., Kunz, J. C., & Law, K. H. (1998). A Hybrid Prescriptive-/Performance-Based Approach to Automated Building Code Checking. *Fifth Congress in Computing in Civil Engineering*, 1–12. Retrieved from http://eil.stanford.edu/publications/chuck_han/9810_ICC.pdf

Hjelseth, E., & Nisbet, N. (2010). Exploring semantic based model checking. *Proceedings of CIB W78 Conference*, 27, 16–18.

Ismail, A. S., Ali, K. N., & Iahad, N. A. (2017). A Review on BIM-based automated code

- compliance checking system. *International Conference on Research and Innovation in Information Systems, ICRIS*. <https://doi.org/10.1109/ICRIS.2017.8002486>
- Khemplani, L. (2005). CORENET e-PlanCheck : Singapore's automated code checking system. *AECbytes - Building the Future, October*, 1–8. Retrieved from <http://www.aecbytes.com/buildingthefuture/CORENETePlanCheck.htm>
- Kim, H., Lee, J.-K., Shin, J., & Choi, J. (2018). Visual Language Approach to Representing KBimCode-based Korea Building Code Sentences for Automated Rule Checking. *Journal of Computational Design and Engineering*. <https://doi.org/10.1016/j.jcde.2018.08.002>
- Lawrence, M., Pottinger, R., Staub-French, S., & Nepal, M. P. (2014). Creating flexible mappings between Building Information Models and cost information. *Automation in Construction*, *45*, 107–118. <https://doi.org/10.1016/j.autcon.2014.05.006>
- Lee, H., Lee, J. K., Park, S., & Kim, I. (2016). Translating building legislation into a computer-executable format for evaluating building permit requirements. *Automation in Construction*, *71*, 49–61. <https://doi.org/10.1016/j.autcon.2016.04.008>
- Lee, H., Lee, S., Park, S., & Lee, J. (2015). An Approach to Translate Korea Building Act into Computer-readable Form for Automated Design Assessment. *Proceedings of the 32nd ISARC*, 1–8.
- Macit İlal, S., & Günaydın, H. M. (2017). Computer representation of building codes for automated compliance checking. *Automation in Construction*, *82*(June), 43–58. <https://doi.org/10.1016/j.autcon.2017.06.018>
- Malsane, S., Matthews, J., Lockley, S., Love, P. E. D., & Greenwood, D. (2015). Development of an object model for automated compliance checking. *Automation in Construction*, *49*(PA), 51–58. <https://doi.org/10.1016/j.autcon.2014.10.004>
- Martins, J. P., & Monteiro, A. (2013). LicA: A BIM based automated code-checking application for water distribution systems. *Automation in Construction*, *29*(23), 12–23.

<https://doi.org/10.1016/j.autcon.2012.08.008>

Melzner, J., Zhang, S., Teizer, J., & Bargstädt, H. J. (2013). A case study on automated safety compliance checking to assist fall protection design and planning in building information models. *Construction Management and Economics*, 31(6), 661–674. <https://doi.org/10.1080/01446193.2013.780662>

Mike, A., Automated, J. S., & Drogemuller, R. (2004). QUT Digital Repository: <http://eprints.qut.edu.au/27228> CONFERENCE THEME: VISUALISATION AND INFORMATION Full Paper, (October), 25–27.

Nawari, N. (2012). The Challenge of Computerizing Building Codes in a BIM Environment. *Computing in Civil Engineering (2012)*, 1, 285–292. <https://doi.org/10.1061/9780784412343.0036>

Nawari, N. O. (2012). BIM-Model Checking in Building Design. *Structures Congress 2012*, 941–952. <https://doi.org/10.1061/9780784412367.084>

Nawari, O., & Email, A. (2011). Automating Codes Conformance in Structural Domain, 569–577.

Nepal, M. P., Staub-French, S., Pottinger, R., & Webster, A. (2012). Querying a building information model for construction-specific spatial information. *Advanced Engineering Informatics*, 26(4), 904–923. <https://doi.org/10.1016/j.aei.2012.08.003>

Park, S., Lee, H., Lee, S., Shin, J., & Lee, J.-K. (2015). Rule checking method-centered approach to represent building permit requirements. *32nd International Symposium on Automation and Robotics in Construction and Mining: Connected to the Future, Proceedings*, 529. Retrieved from https://www.engineeringvillage.com/share/document.url?mid=cpx_M4a1f008415429f11f12M469c10178163171&database=cpx

Pauwels, P., Van Deursen, D., Verstraeten, R., De Roo, J., De Meyer, R., Van De Walle, R., &

- Van Campenhout, J. (2011). A semantic rule checking environment for building performance checking. *Automation in Construction*, 20(5), 506–518. <https://doi.org/10.1016/j.autcon.2010.11.017>
- Philosophy Index. (2018). No Title. Retrieved May 9, 2018, from <http://www.philosophy-index.com/logic/symbolic/>
- Poças Martins, J. P., & Abrantes, V. (2010). Automated code-checking as a driver of BIM adoption. *International Journal for Housing Science and Its Applications*. <https://doi.org/10.1017/CBO9781107415324.004>
- Practices, B. (n.d.). Building Framing Systems, 1–20.
- Preidel, C., & Borrmann, A. (2015). Automated Code Compliance Checking Based on a Visual Language and Building Information Modeling. *Proceedings of the 32nd International Symposium of Automation and Robotics in Construction, 15-18 June*, 256–263. <https://doi.org/10.13140/RG.2.1.1542.2805>
- Rasdorf, B. W. J., & Lakmazaheri, S. (1990). LOGIC - BASED APPROACH FOR MODELING ORGANIZATION OF DESIGN STANDARDS In general , a design standard is composed of a set of provisions , where each provision defines a set of rules (constraints) that have to be satisfied in a give, 4(2), 102–123.
- Rhinoceros3D., website. (2018). Retrieved from <https://www.rhino3d.com/>
- See, R. (2008). SMARTcodes : Enabling BIM Based Automated Code Compliance Checking AEC-ST Conference Presentation – 21-May-08 SMARTcodes : Enabling BIM Based Automated Code Compliance Checking, (October 2007), 1–7.
- Shih, S.-Y., Sher, W., & Giggins, H. (2013). Assessment of the building code of Australia to inform the development of BIM-enabled code checking system. *Proceedings of the 19th World Building Congress: Construction and Society, 5-9 May*, 1–12. Retrieved from <http://hdl.handle.net/1959.13/1050708>

- Sjøgren. (2007). in Norway.
- Solihin, W., & Eastman, C. (2015). A Knowledge Representation Approach to Capturing BIM Based Rule Checking Requirements Using Conceptual Graph. *Proc. of the 32nd CIB W78 Conference 2015, 27th-29th October 2015, Eindhoven, The Netherlands*, 686–695.
- Solihin, W., & Eastman, C. (2015). Classification of rules for automated BIM rule checking development. *Automation in Construction*, 53, 69–82.
<https://doi.org/10.1016/j.autcon.2015.03.003>
- STATSBTGG website. (2018). Retrieved from <https://www.statsbygg.no/>
- Tan, X., Hammad, A., & Fazio, P. (2010). Automated Code Compliance Checking for Building Envelope Design. *Journal of Computing in Civil Engineering*, 24(2), 203–211.
[https://doi.org/10.1061/\(ASCE\)0887-3801\(2010\)24:2\(203\)](https://doi.org/10.1061/(ASCE)0887-3801(2010)24:2(203))
- Terkaj, W., & Šojić, A. (2015). Ontology-based representation of IFC EXPRESS rules: An enhancement of the ifcOWL ontology. *Automation in Construction*, 57, 188–201.
<https://doi.org/10.1016/j.autcon.2015.04.010>
- Uhm, M., Lee, G., Park, Y., Kim, S., Jung, J., & Lee, J. K. (2015). Requirements for computational rule checking of requests for proposals (RFPs) for building designs in South Korea. *Advanced Engineering Informatics*, 29(3), 602–615.
<https://doi.org/10.1016/j.aei.2015.05.006>
- Yang, Q. Z., & Xu, X. (2004). Design knowledge modeling and software implementation for building code compliance checking. *Building and Environment*, 39(6), 689–698.
<https://doi.org/10.1016/j.buildenv.2003.12.004>
- Zhang, J., & El-Gohary, N. M. (2017). Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction*, 73, 45–57. <https://doi.org/10.1016/j.autcon.2016.08.027>
- Zhang, S., Teizer, J., Lee, J. K., Eastman, C. M., & Venugopal, M. (2013). Building

Information Modeling (BIM) and Safety: Automatic Safety Checking of Construction Models and Schedules. *Automation in Construction*, 29, 183–195.

<https://doi.org/10.1016/j.autcon.2012.05.006>

Zhou, W., Whyte, J., & Sacks, R. (2012). Construction safety and digital design: A review.

Automation in Construction, 22, 102–111. <https://doi.org/10.1016/j.autcon.2011.07.005>

APPENDIX A: Glossary

Abut or **abutting** - immediately contiguous to or physically touching, and when used with respect to a lot or Site, means that the lot or Site physically touches upon another lot, Site, or piece of land, and shares a property line or boundary line with it.

Setback distance - the minimum distance between the edge of property and where the structure will be built, the distances with respect to orientation of building are called front setback, side setback, and rear setback.

Corner Lot - a lot situated at the intersection of two or more streets having an interior angle of intersection of 135 degrees or less, or, where one street bends to create an interior angle of 135 degrees or less.

Lane - an alley as defined in the Highway Traffic Act, 1980.

Zone - specific group of listed Use Classes and Development Regulations, which regulate the use, and development of land within specific geographic areas of the City. The Use Classes and Development Regulations are contained in Parts II and IV of this Bylaw and may be subject to the regulations contained in Part I of this Bylaw, while the geographic areas to which they apply are shown on the Zoning Map, comprising Part III of the Bylaw.

APPENDIX B: User Manual

User manual for DCheck add-on application software,

The step by step process for using the add-on application is as follows:

Step 1. The residential building to be checked for the lot design in according to Edmonton zoning by laws and framing design according to Alberta building code should be first designed in Autodesk Revit 2017 with level of details more than LOD 300,

Step 2. Property lines should be drawn with respect to site measurements and framing of the building model should be done.

Step 3. After completing designing the model, click on the DCheck button in top ribbon in Revit application, then by clicking on project data button an user interface windows will pop-up as shown in the figure 1.

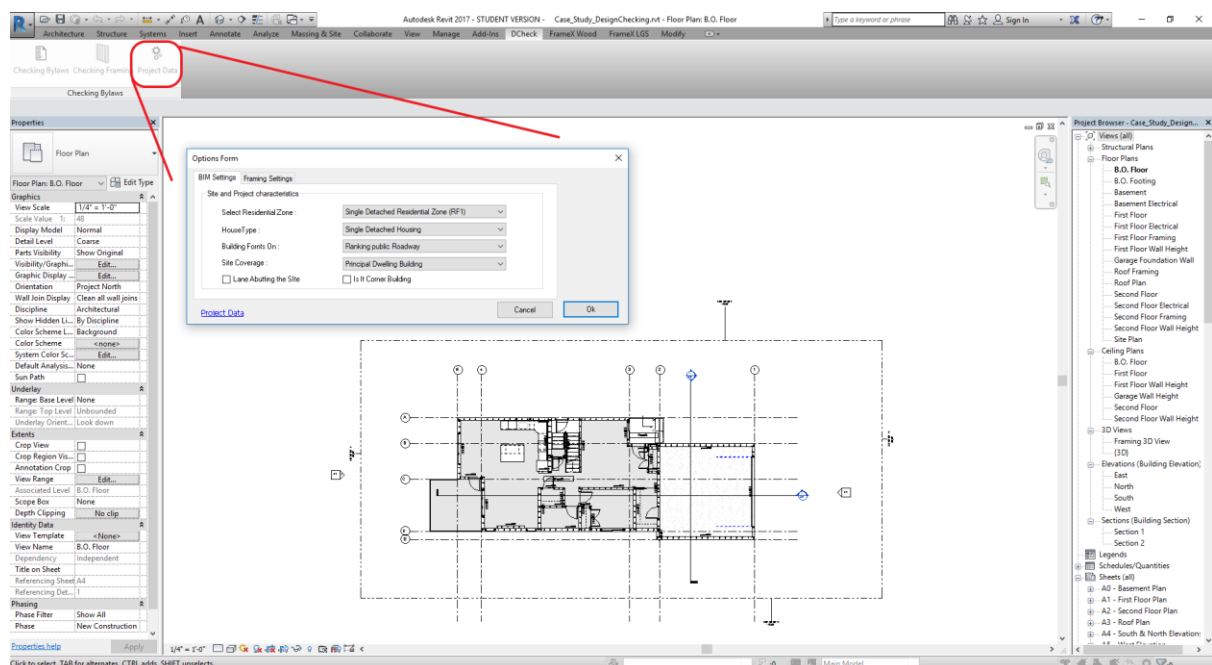


Figure 1. User interface of DCheck application.

Step 4. User needs to provide the information related to Edmonton zoning and the wood framing, that is user needs to select zone where the house is being built from a drop

down menu with list of ten different zones, that are Single Detached Residential Zone (RF1), Residential small Lot Zone (RSL), Low Density Infill Zone (RF2), Planned Lot Residential Zone (RPL), Small Scale Infill Development Zone (RF3), Semi-detached Residential Zone (RF4), Residential Mixed Dwelling Zone (RMD), Row Housing Zone, Urban Character Row Housing Zone (UCRH), Medium Density Multiple Family Zone (RF6).

Figure 2. Main user interface

Select the residential house intended to build in those zones from drop down list of single-detached housing (sdh), semi-detached housing (ssh), duplex housing (dh), limited group homes (lgh), garden suite (gs), secondary suites (ss), and minor home-based business (mhb).

Select the building fronts on condition and other zoning conditions like lane abutting site or not. And is it a corner site or not which are related to residential building site

conditions.

Step 5. By clicking on framing setting, provide some information regarding framing, like type of lumber material grade. Load details like live load, dead load and snow load on the residential building.

Check mark on some condition if present depending on design of residential building, that are whether attic is accessible stairs or not. Roof with storage area.

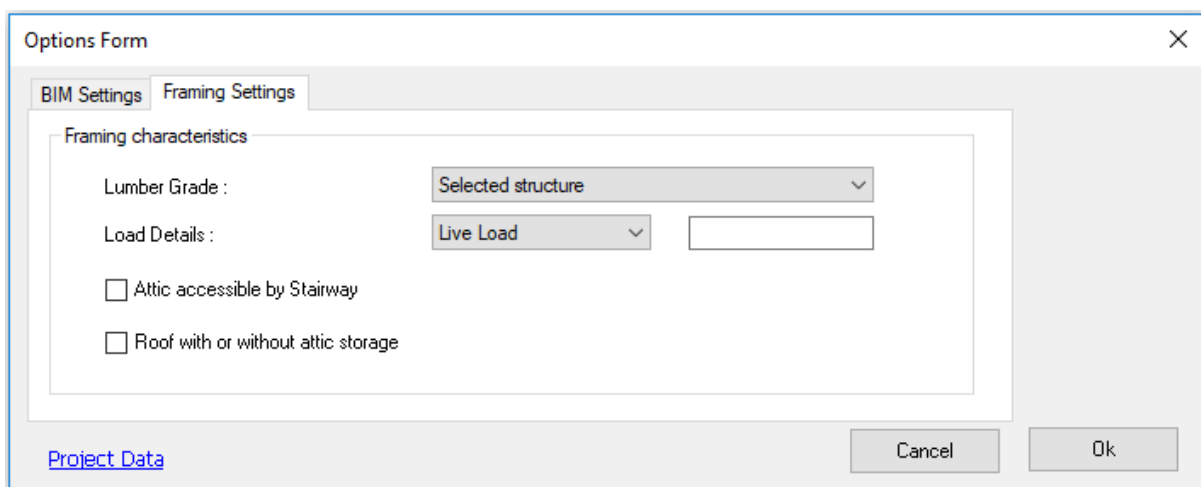


Figure 3. interface to provide wood framing details.

Step 6. Provide all the details related to architect, builder and owner of the project and site location details with respect to Edmonton zoning, all this information's are same that needs to be provided even with the submission of 2D CAD drawings manually. And then press Ok.

Step 7. By clicking on Check Bylaws button, the application will run in background and windows with bylaw checking related to lot dimensions in accordance with Edmonton municipal bylaws will be provided in text format.

Which provides the user with all the error's present in the lot design in BIM model

designed. Each error will be provided with reason for failure and information have that can be corrected to compile with bylaws.

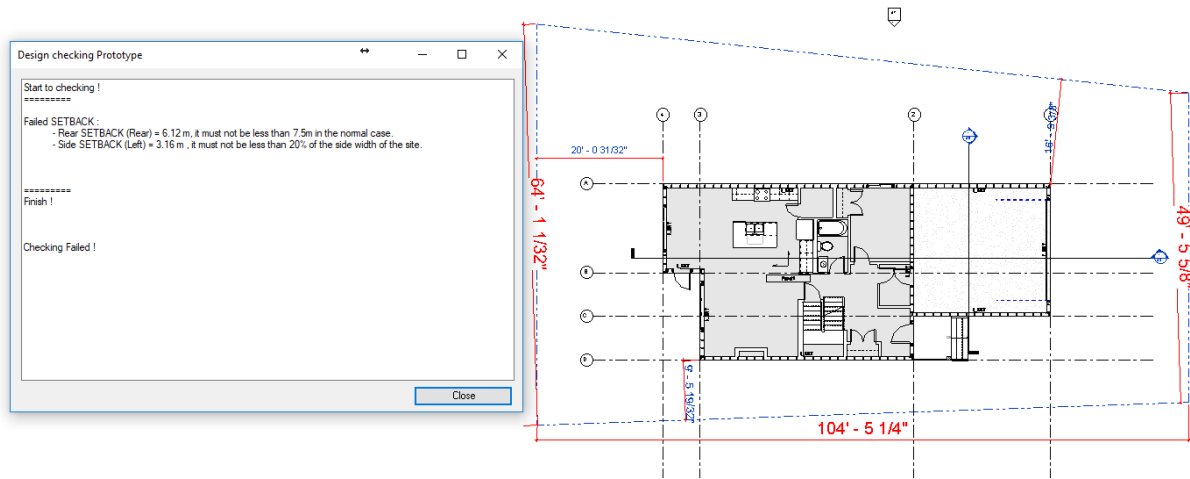


Figure 4. Bylaws check results window showing errors present in design.

Step 8. Press Ok. And then by clicking on Framing Check button in top ribbon, that will pop up the windows with checking report for the framing design of model.

The report will provide reason for failure and information have it can be corrected, by clicking on select objects button in windows that will highlights all the error related objects in the building model, for easy identification of errors and can be corrected.

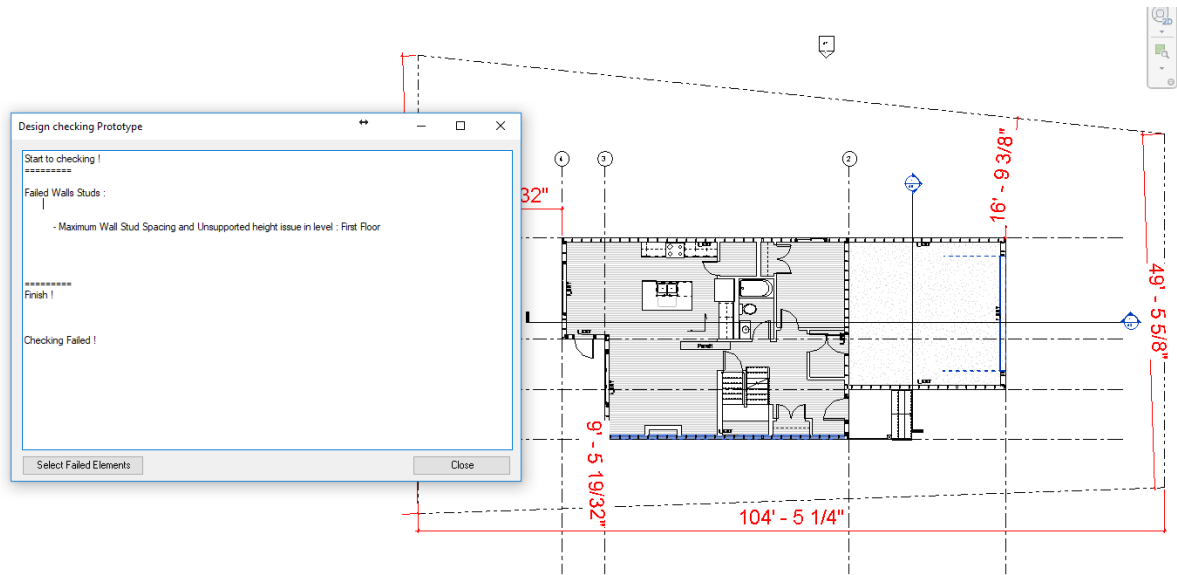


Figure 5. Framing check results window showing errors present in design.

Step 9. After correcting all the errors present in design according to information provided in the report, then again by clicking on bylaw Check and framing Check button, application will run and provide a text report if further any errors present in redesigned model.

Step 10. If there were no errors present in the designed model, then a report with text, check successful message will be shown as below figure.

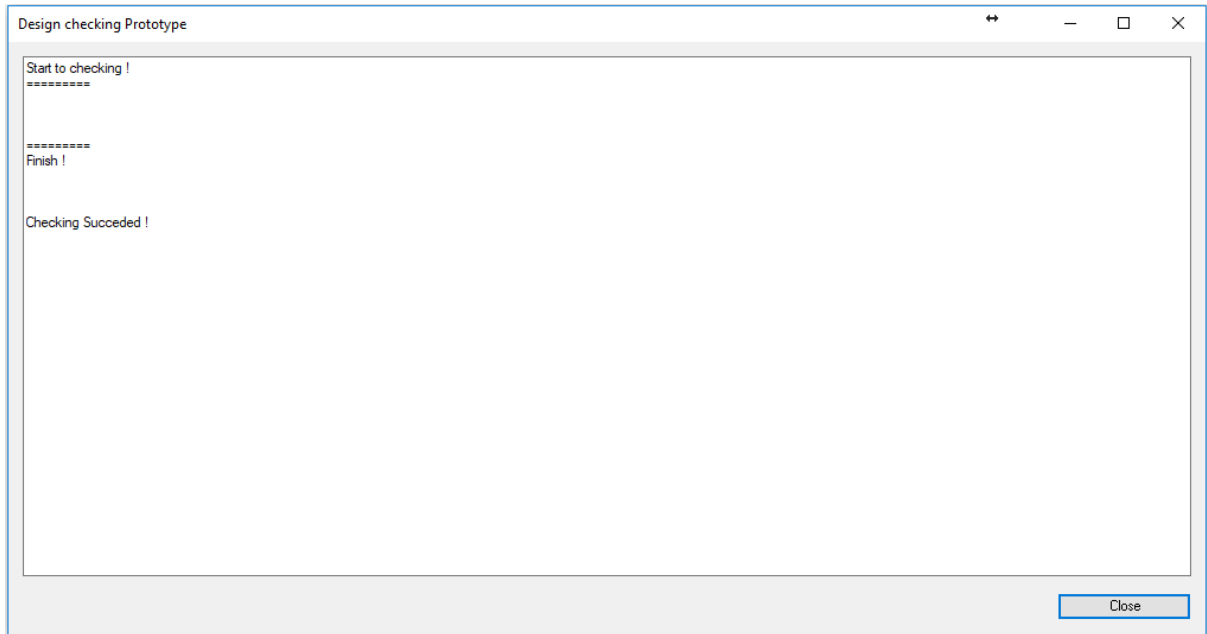


Figure 6. Final check results without any errors in design.

APPENDIX C: All the Regulations Related to Framing from Building Code

Table 5. Framing Regulations from Alberta Building Code 2014.

Object	Conditions	Operator	Threshold Value
Studs	Small repetitive structural members are spaced.	Maximum.	600 mm. o.c.,
	Span of any structural member (except post construction, beam, and plank construction and log construction).	Maximum.	12.20 m.
	Top plates in walls shall not be notched, drilled or reduced the undamaged width to.	Minimum.	50 mm.
	Ends of wood joist, beams, and other members framing into masonry or concrete where bottom of member is below GL – should be treated to prevent decay or provide space at end or side.	Minimum.	12 mm.
Backing lumber	Non-load bearing walls should be supported on.		Floor joist. / on backing b/w joist.
	Backing lumber shall be.	Minimum.	38x89 mm. @ Max. 1.2 m. apart.
	Interior Load bearing walls parallel to joist should be supported by Beams/ walls Unless joists are designed to support.		900mm. if that wall NOT supports floor.
	Loadbearing interior walls perpendicular to floor joist shall be located within from joist support.	Maximum.	OR 600mm. if that wall supports floor.
	All studs should be placed at.		right angle (90°) to the wall face.
	Except, wall studs support only load from attic not accessible by a stairway are permitted where studs are clad on at least on side with plywood, OSB or waterboard fastened and portion of roof supported by studs do not exceed 2.1m. in width. Can be place.		flat (0°).
Exterior corners should have at least.	Minimum.	2 studs.	
Size of wall plates thickness.	Minimum.	38 mm.	

Top/bottom plates	<p>If studs are located directly -thickness. -width</p> <p>Studs on sides of openings. If lintel opening span is more than 3m. long - studs should be tripled on each side.</p>	<p>Minimum. Minimum.</p>	<p>19 mm. Width of wall stud. 2 from bottom of lintel to top of bottom wall plate, & 1 from bottom of top wall plate to bottom wall pate.</p>
	<p>If lintel opening span is less than 3m. long - double studs can be placed on each side.</p>	<p>Equal.</p>	<p>1 from bottom of lintel to top of bottom wall plate, & 1 from bottom of top wall plate to bottom wall pate.</p>
	<p>For non-loadbearing interior walls and walls not require fire rating.</p>	<p>Minimum.</p>	<p>single studs on both sides.</p>
	<p>Opening is less than or equal to required stud spacing and no such two openings are in adjacent spaces.</p>		<p>single studs on both sides.</p>
	<p>Bottom plate should be provided in all the cases.</p>	<p>At least.</p>	<p>1</p>
	<p>Bottom plate should not project more than.</p>	<p>Minimum.</p>	<p>1/3 the plate width over the support</p>
	<p>Top plats in load bearing wall.</p>	<p>Equal.</p>	<p>2 plates shall be provided.</p>
	<p>Except, wall contain a lintel provided the top plate forms a tie across the lintel.</p> <p>Top plate can be not provided for load bearing wall - if lintel is tied to the adjacent wall section with.</p>	<p>Minimum.</p>	<p>1 top plate. 75×150×0.91 mm thick galvanized steel. OR</p>

			19×89×300 mm wood splice nailed to each wall section with Min. 63mm nails.
	Framing over openings. In non-loadbearing walls - for normal wall.	Minimum.	38 mm thick with same width as stud.
	- for fire resistance wall.	Minimum.	2, 38 mm thick with same width of studs.
Studs	Notching or drilling of studs.	Minimum.	1/3 of stud depth.
	If more than that, with 2 in. lumber nailed to the sides of the stud and extending at least 40mm. and if only solid wood remains. same for load bearing walls too, at least 2/3 of solid portion of stud should be remaining or else it should be.	Minimum.	600 mm. (24 in.) on each side.
	Partition wall if it does not contain a swinging door.	Minimum.	2×4 nominal studs at 16 in. c/c with wide face of stud parallel to the wall.
	Single studs can be used at door openings in partition wall and can have single top plate with.	Minimum.	2 in. thick lumber with same width as the studs.
	Floor, roof, and ceiling framing members are permitted to be notched, that notch is located within half the joist depth from edge and is not deeper than.	Minimum.	1/3 of depth of member.
Floor Height	Wall studs which are load bearing or 40 mm. - load bearing should be not damaged, drilled or if done damaged portion should be.	Minimum.	1/3 of depth of member.
	Clear Height. Ceiling height for secondary suite.	Minimum.	1.95 m.
	Under beams and ducting in secondary suite.	Minimum.	1.85 m.

	Other than above ceiling Height should be.	Minimum.	2.1 m.
	Storage garage.	Minimum.	2.0 m.
	Clear height in exit and access to exit (except for stairways, doorways, and storage garage).	Minimum.	2.1 m.
	Clear height in exits and access to exit in storage garages.	Minimum.	2 m.
	For Residential (group C) buildings can have stories up to.	Maximum.	3.
	All other occupancies can have stories up to.	Maximum.	2/3.
	Roof space or attic should be provided with opening where open space in attic measures -Area. -Length/width. -Height.	Equal.	3 m ² 1 m. or more. 600 mm.
Openings.	Opening for attic should be.	Minimum.	550 × 900 mm.
	Opening can be less than above when Area less than. And all dimensions.	Equal. Minimum.	0.32 m ² 500 mm.
Roof	Roof truss members. or otherwise weakened unless such notching is allowed in design of truss.		shall not be notched, drilled.
	Roof and ceiling framing members openings greater than 2 rafter or joist spacings wide.		shall be doubled on each side of openings.
	Roof truss if not designed according to part 4, should support.	Minimum.	Ceiling load (DL+LL) of 0.35kPa + 2/3 times the specific live roof load for 24 hr.
	When a compression web member in roof trusses exceed provided with continuous bracing buckling.	At least.	19 × 89 mm. nailed at right to web with 2, 63 mm nails for each member.

	A roof that slop is less than 1:3 should be vertical supported at the peak with.		2×6 ridge beam supported at 1.2 m (4 ft.) vertical struts.
	Gable-end projection extending more than 300 mm.(12 inches) Beyond the wall should be supported.		By framing members called lookouts.
Floor.	Crawl space, an access opening for -single dwelling unit. -for other units.	Minimum. Minimum.	500×700 mm. 550×900 mm.
	Live load on subfloors and floor framing.	Minimum.	mm.
	Holes drilled in roof, floor or ceiling framing members shall be.	Minimum.	2.4 kPa
	And holes should be located, unless member depth increased by size of hole.	Minimum.	1/4 of depth of member. 50 mm from edges.
	Joist supported by beam when connected on sides with - with ledger strip nailed to side of beam. - with ledger strip nailed to side of beam with at least.	Minimum. Minimum.	38 × 64 mm. lumber. 38 × 38 mm. Four 89 mm dia. Bolts
Strapping.	Strapping in joist should be with	Minimum.	19×64mm. lumber, nailed at bottom of joist.
	Distance from support or from other strapping located not more than.	Maximum.	2,100 mm.
	Bridging in joist should be with	Minimum.	19×64 mm /38×38 mm and nailed at bottom of joist.
	from support or from other strapping located not more than.	Maximum.	2,100 mm.
	strapping not required if furring strip are fastened directly to the joist or a panel-type ceiling finish complying directly to the joists.		12.7 mm thick. 19×89 mm @ Max. 600

Joist	When ceiling is attached to wood furring - ceiling finish gypsum board, plywood or OSB. - furring shall be.		mm o.c., OR 19×64 mm @ Max. 400 mm o.c.,
	Header joist around opening should be doubled if opening span.	Maximum.	1.2 m.
	Trimmer joist around floor opening shall be doubled if length of header joist.	Greater than.	800 mm
	If header joist greater than 3 m for header, 2 m. for trimmer in length. then size of header should be decided by calculations.	Maximum.	400 mm. with 38×184 mm joist. OR 600 mm with 38×235 mm joist. or more.
	Floor joist supporting roof load can be cantilevered beyond their support. Only for roof load, shall not support floor load. Unless designed.		
	Cantilever floor joist right angle to floor joists, the tail joist in cantilever portion shall extend inward away from support to a distance of.	Minimum.	6 times the length of the cantilever.
	A single top plate may be installed in stud walls provided the plate is adequately tied at joints provided the rafters or joist are centered over the studs with tolerance of.	Minimum.	24 mm.
	Joist end bearing should be at least.	Minimum.	1 ½ in. on wood. 3 in. on masonry.
	Bridging in joist are provided where nominal depth to thickness ratio of joist exceed installed at	Equal.	8 ft intervals.
	Support of partitions - may be off set from supporting members by no more than.	Equal.	Depth of joist
Notches made on the upper lumber joists near their ends must be located within.	Equal.	½ joist depth from support.	
And their depth cannot be more than.	Equal.	1/3 of joist	

Columns	Notches are not permitted on the bottom of the joists.		depth.
	When load bearing wall runs parallel to joist or load bearing wall in the basement.	it must.	be supported by beam.
	Load bearing wall located at right angle to floor joist Should be located not more than		
	- if wall not support a floor.	Maximum.	900 mm. (36 in.) from the joist support.
	- supports one or more floors.	Maximum.	600 mm. (24 in.) from joist support.
	Notches at top of joist, deeper joist must be used so that the net depth at the notch is equal to or greater than.	Equal.	Joist depth.
	Non-load bearing wall parallel to the joist should bear on joist or on backing between joists, backing should be	Minimum.	2×4 in. nominal with spaced at 1.2 m (4 ft) or less on center.
	columns		
	should be for not more than.	Minimum.	2 floors.
	live load not exceeds.	Maximum.	2.4 kPa.
	length of joist not exceed.	Maximum.	5 m.
	sum of snow and occupancy load does not exceed.	Maximum.	4.8 kPa.
	Built-up column shall consist of.	Minimum.	38mm. thick full-length members.
Bolted together with not less than.	Minimum.	9.25 mm. @ Max. 450 mm. o.c.,	
Nailed together with not less than.	Minimum.	76 mm. @ Max. 300 mm. o.c.,	
Wood column should be separated from concrete in contact with ground by.	Equal.	0.05 mm Polyethylene film.	
the width or diameter of column should be	Equal.		
columns for garages and carports	Minimum.	Width of supported member.	
- round column	Maximum.		
- rectangular column		184 mm.	

Doors	- Except above columns can be.		140×140mm. 89×89 mm.
	In plank and beam framing method beams of adequate size is spaced up to.		
	Joist can rest on top of beam in which case the top of beam is level with the top of the sill plate. This joist should lap above the beam and the recommended length of lap.	Maximum.	8 ft 300 mm. (12 in).
	House with secondary suite, entrance door width. door height.	Minimum. Minimum.	810 mm. 1980 mm.
	All doors in at least one line of passage from exterior to the basement utility rooms. door width. door height.	Minimum. Minimum.	810 mm. 1980 mm.
	Bathroom, water-closet room shower, doors located off hallways. door width. door height. (for secondary suite shower room) Door height.	Minimum. Minimum. Minimum.	610 mm. 1980 mm. 1890 mm.
	Doorways to public water-closet rooms. door width. door height.	Minimum. Minimum.	810 mm. 2030 mm.
	Except for doors and corridors width of every exit. door width.	Minimum.	900 mm.
	Clear height of doorway (for exit doors, doors open into or located within public corridor, and doors open into or located in facility that provides access to exit a suite). door height. door width.(if only one leaf door). door width. (if multi-leaf doors are installed with active leaves).	Minimum. Minimum. Minimum.	2030 mm. 800 mm. 1210 mm.
	exit stairs for single or with secondary suite – Width	Minimum.	860 mm.
except above mentioned for residential homes – width	Minimum.	900 mm.	
exit stairs and public stairs serving other than residential/ 8 mm. per person. – width	Minimum.	900 mm.	

Stairs Landing	clear height over stairs for single or with secondary suite including their common space – height.	Minimum.	1950 mm.
	under beams and ducting in secondary suites – height	Minimum.	1850 mm.
	except for stairs in dwelling unit, shall be provided with.		at least 3 risers in interior flight.
	vertical height of any flight of stairs shall not exceed.		3.7 m.
	dimensions for risers (nosing to nosing):	Maximum.	200 mm.
	- Private	Minimum.	125 mm.
	- Public	Maximum.	180 mm.
		Minimum.	125 mm.
	dimensions for tread	Maximum.	335 mm.
	- Private	Minimum.	210 mm.
	- Public	Maximum.	No Limit
		Minimum.	280 mm.
	Angled tread	Minimum.	200 mm.
	tread dimension	Maximum.	Run dimension + 25 mm.
uniformity and tolerance for risers and treads.	Equal.	5 mm.	
- b/w adjacent tread and landing.		10 mm.	
- b/w tallest and shortest risers in flight.	Maximum.	1 in 50	
slop of treads should not exceed.	Equal.	30° to 45°	
winders angle should be in between.	Maximum.	90°	
when winders incorporated into stair, each set shall not turn through more than.			
spiral stairs are used if	maximum.	230 mm.	
- rise is.	More than.	140 mm.	
- average run.	Less than.	660 mm.	
- width of flight.			
landing dimensions			
For single dwellings - in straight run or landing			

Decks	turning through less than 30° within dwelling unit.	Minimum.	Width of stair.
	- Width.	Minimum.	860 mm.
	- Length.		
	For exterior stair.	Minimum.	Width of stair.
	- Width.	Minimum.	900 mm.
	- Length.		
	Landing turning through an angle between 30° to 90.	Minimum.	230 mm.
	- Width. (measured at inside edge of landing)	Minimum.	370 mm.
	- Length. (measured from outside of the landing)		
	Landing turning through an angle more than 90°	Minimum.	Width of stair.
	- Width.	Minimum.	Width of stair.
	- Length.		
	For other than single dwellings - in straight run or landing turning through less than 30°	Minimum.	Width of stair.
	- Width.	Minimum.	Lesser of width of stairs/1100 mm.
- Length.			
Landing turning through 30° or more.	Minimum.	Width of stair at right angle to path of travel.	
- Width.		Width of stair.	
		1950 mm	
Clear height over landing in single dwelling or A house with secondary suite – Height.	Minimum.	2050 mm	
Except as permitted above – height.	Minimum.	25 mm	
wood stringers – thickness.	Minimum.	900 mm high.	
Decks, Porches and balconies over 600 mm. and less than 1800 mm. above the finished ground level are required have guard.	at least.	1070 mm high.	
Greater than 1800 mm. should have.	at least.	32 mm. (1 ¼ in.) @ 16 in centers OR	
Decking thickness should be at least.	Minimum.	38 mm. @	

	Low level decks are built at least	Minimum.	24 in centers. 150 mm (6 in.) from ground.
--	------------------------------------	----------	---

Table 6. Maximum spans for floor joist table

SPF(Spruce-Pine-Fir)

grade	Joist size, mm.	Joist spacing with strapping, m			Joist spacing with bridging, m			With strapping and bridging, m		
		300	400	600	300	400	600	300	400	600
Selected structural	38×89	1.95	1.81	1.64	2.06	1.87	1.64	2.06	1.87	1.64
	38×140	3.05	2.85	2.57	3.24	2.95	2.57	3.24	2.95	2.57
	38×184	3.66	3.48	3.31	3.94	3.70	3.38	4.12	3.84	3.38
	38×235	4.31	4.10	3.90	4.59	4.31	4.05	4.76	4.44	4.14
	38×286	4.91	4.67	4.45	5.18	4.87	4.57	5.34	4.98	4.64
No. 1 and No.2	38×89	1.86	1.72	1.58	1.99	1.81	1.58	1.99	1.81	1.58
	38×140	2.92	2.71	2.49	3.14	2.85	2.49	3.14	2.85	2.49
	38×184	3.54	3.36	3.20	3.81	3.58	3.27	3.99	3.72	3.27
	38×235	4.17	3.96	3.77	4.44	4.17	3.92	4.60	4.29	4.00
	38×286	4.75	4.52	4.30	5.01	4.71	4.42	5.17	4.82	4.49
No. 3	38×89	1.81	1.68	1.55	1.96	1.78	1.55	1.96	1.78	1.55
	38×140	2.84	2.64	2.43	3.08	2.80	2.43	3.08	2.80	2.43
	38×184	3.47	3.30	2.95	3.74	3.52	2.95	3.92	3.61	2.95
	38×235	4.09	3.89	3.61	4.36	4.09	3.61	4.52	4.22	3.61
	38×286	4.67	4.44	4.19	4.92	4.62	4.19	5.08	4.73	4.19
Construction	38×89	1.81	1.68	1.55	1.96	1.78	1.55	1.96	1.78	1.55
Standard	38×89	1.70	1.58	1.44	1.88	1.71	1.44	1.88	1.71	1.44

APPENDIX D: Extended Data Structure Developed for Prototype Excerpt from the C#.Net

Excerpt from the C#.Net codes for Extended data structure developed for deriving some attribute values in DCheck platform:

```
using Autodesk.Revit.DB;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Checking_Prototype
{
    public static class RevitExtensions
    {
        // for getting width of property line
        public static double GetWidth(this BoundingBoxXYZ bBox)
        {
            return bBox.Max.Y - bBox.Min.Y;
        }

        // for getting depth of property line
        public static double GetDepth(this BoundingBoxXYZ bBox)
        {
            return bBox.Max.X - bBox.Min.X;
        }

        // getting face details of object
        public static List<Face> GetFaces(this Element element)
        {
            var faces = new List<Face>();

            var geom = element.get_Geometry(new Options());

            foreach (GeometryObject item in geom)
            {
                if (item is Solid)
```

```

    {
        var solid = item as Solid;
        foreach (Face f in solid.Faces)
            faces.Add(f);
    }
    else if (item is GeometryInstance)
    {
        var g = item as GeometryInstance;
        var model = g.GetInstanceGeometry();
        foreach (var g1 in model)
        {
            if (g1 is Solid)
            {
                var solid = item as Solid;
                foreach (Face f in solid.Faces)
                    faces.Add(f);
            }
        }
    }
}
return faces;
}
//for knowing the slop of roof
public static bool IsSlopedFacein3D(this Face face)
{
    var triangle = face.Triangulate().get_Triangle(0);
    var pt1 = triangle.get_Vertex(0);
    var pt2 = triangle.get_Vertex(1);
    var pt3 = triangle.get_Vertex(2);
    var vec1 = pt1 - pt2;
    var vec2 = pt3 - pt2;
    var normalVec = vec1.CrossProduct(vec2).Normalize();
}

```

```

        if (normalVec.Z.IsApproxEqual(0) || (normalVec.X.IsApproxEqual(0) &&
normalVec.Y.IsApproxEqual(0)))
            return false;
        return true;
    }
    public static bool IsGableRoof(this FootPrintRoof roof)
    {
        var faces = roof.GetFaces();
        if (faces.Any(o => o.IsSlopedFacein3D()))
            return true;
        return false;
    }
    public static bool IsParralel(this Curve c1, Curve c2)
    {
        var dx1 = c1.GetEndPoint(1).X - c1.GetEndPoint(0).X;
        var dy1 = c1.GetEndPoint(1).Y - c1.GetEndPoint(0).Y;
        var dx2 = c2.GetEndPoint(1).X - c2.GetEndPoint(0).X;
        var dy2 = c2.GetEndPoint(1).Y - c2.GetEndPoint(0).Y;
        var cosAngle = Math.Abs((dx1 * dx2 + dy1 * dy2) / Math.Sqrt((dx1 * dx1 +
dy1 * dy1) * (dx2 * dx2 + dy2 * dy2)));
        if (cosAngle > 0.1)
            return true;
        return false;
    }
    public static bool IsOverlapTotaly(this Curve c1, Curve c2)
    {
        var c1_2D = c1.ToLine2D();
        var c2_2D = c2.ToLine2D();
        var pt1 = c1_2D.GetEndPoint(0);
        var pt2 = c1_2D.GetEndPoint(1);
        var proj1 = c2_2D.GetEndPoint(0).GetProjection(c1_2D);

```

```

        var proj2 = c2_2D.GetEndPoint(1).GetProjection(c1_2D);

        if ((proj1.IsAlmostEqualTo(pt1) || proj1.IsAlmostEqualTo(pt2)) &&
(proj2.IsAlmostEqualTo(pt1) || proj2.IsAlmostEqualTo(pt2)))

            return true;

        return false;
    }

    public static bool ArePointsInSameSide2D(this Curve line, XYZ pt1, XYZ pt2)
    {
        var line2D = line.ToLine2D();
        var pt1_2D = pt1.ToPoint2D();
        var pt2_2D = pt2.ToPoint2D();

        var vec1 = (pt1_2D.GetProjection(line2D) - pt1_2D).Normalize();
        var vec2 = (pt2_2D.GetProjection(line2D) - pt2_2D).Normalize();
        if (vec1.IsAlmostEqualTo(vec2))

            return true;

        return false;
    }

    public static double DistanceToProjection2D(this XYZ pt, Curve line)
    {
        var pt2D = new XYZ(pt.X, pt.Y, 0);
        var line2D = Line.CreateBound(line.GetEndPoint(0).ToPoint2D(),
line.GetEndPoint(1).ToPoint2D());

        return (pt2D.GetProjection(line2D) - pt2D).GetLength();
    }

    public static XYZ ToPoint2D(this XYZ pt)
    {
        return new XYZ(pt.X, pt.Y, 0);
    }

    public static Curve ToLine2D(this Curve line)
    {

```

```

        return Line.CreateBound(line.GetEndPoint(0).ToPoint2D(),
line.GetEndPoint(1).ToPoint2D());
    }
    //to get edge distance
    public static XYZ GetProjection(this XYZ pt, Curve line)
    {
        var start = line.GetEndPoint(0);
        var end = line.GetEndPoint(1);
        var v1 = pt - start;
        var v2 = end - start;
        var l2 = v2.X * v2.X + v2.Y * v2.Y;
        var dot = v1.DotProduct(v2);
        var nDist = dot / l2;
        return new XYZ(start.X + v2.X * nDist, start.Y + v2.Y * nDist, start.Z +
v2.Z * nDist);
    }
    public static XYZ GetMidPoint(this Curve line)
    {
        return line.GetEndPoint(0) + (line.GetVector().Normalize() * 0.5 *
line.Length);
    }
    public static bool IsLeft(this Curve line,XYZ pt)
    {
        var vec = line.GetVector().ToPoint2D();
        var normVec = new XYZ(-vec.Y, vec.X,0).Normalize();
        var ptVec = (pt - pt.GetProjection(line)).Normalize();
        if (normVec.IsAlmostEqualTo(ptVec))
            return true;
        return false;
    }
}

```

```

public static XYZ GetVector(this Curve line)
{
    return line.GetEndPoint(1) - line.GetEndPoint(0);
}

//getting boundaries of rectangle
public static List<Curve> GetRectangleBounds(this BoundingBoxXYZ bbox)
{
    var minX = bbox.Min.X;
    var maxX = bbox.Max.X;
    var minY = bbox.Min.Y;
    var maxY = bbox.Max.Y;
    var list = new List<Curve>();
    list.Add(Line.CreateBound(new XYZ(minX, minY, 0), new XYZ(maxX, minY, 0)));
    list.Add(Line.CreateBound(new XYZ(maxX, minY, 0), new XYZ(maxX, maxY, 0)));
    list.Add(Line.CreateBound(new XYZ(maxX, maxY, 0), new XYZ(minX, maxY, 0)));
    list.Add(Line.CreateBound(new XYZ(minX, maxY, 0), new XYZ(minX, minY, 0)));
    return list;
}

public static bool IsApproxEqual(this double num1, double num2)
{
    return Math.Abs(num1 - num2) < 0.000001;
}

public static XYZ GetNormalVec(this Face face)
{
    var mesh = face.Triangulate();
    var tra1 = mesh.get_Triangle(0);
    var pt0 = tra1.get_Vertex(0);
    var pt1 = tra1.get_Vertex(1);
    var pt2 = tra1.get_Vertex(2);
    var vec1 = pt1 - pt0;
    var vec2 = pt2 - pt0;
}

```

```

        return vec1.CrossProduct(vec2).Normalize();
    }
    public static double GetNormalDistanceBetween(this Curve c1, Curve c2)
    {
        return c1.GetEndPoint(0).DistanceToProjection2D(c2);
    }
    public static bool Contains(this Curve line, XYZ pt)
    {
        var dist1 = pt.DistanceTo(line.GetEndPoint(0));
        var dist2 = pt.DistanceTo(line.GetEndPoint(1));
        if (Math.Abs(line.Length - (dist1 + dist2)) < 0.0001)
            return true;
        return false;
    }
    public static Curve GetCurve(this Element elem)
    {
        return (elem.Location as LocationCurve).Curve;
    }
    public static XYZ GetCentroid(this BoundingBoxXYZ bBox)
    {
        return bBox.Min + ( bBox.Max - bBox.Min) * 0.5;
    }
    public static void Draw(this Curve c, Document doc)
    {
        using (var t = new Transaction(doc))
        {
            t.Start("Create line");
            doc.Create.NewDetailCurve(doc.ActiveView, c);
            t.Commit();
        }
    }
}
}
}

```