

**A KNOWLEDGE-BASED SYSTEM FRAMEWORK FOR SEMANTIC
ENRICHMENT AND AUTOMATED DETAILED DESIGN IN THE AEC PROJECTS**

A Dissertation
Presented to
The Academic Faculty

by

Shiva Aram

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
College of Architecture

Georgia Institute of Technology

December 2014

**A KNOWLEDGE-BASED SYSTEM FRAMEWORK FOR SEMANTIC
ENRICHMENT AND AUTOMATED DETAILED DESIGN IN THE AEC PROJECTS**

Approved by:

Professor Charles M. Eastman, Advisor
School of Architecture
Georgia Institute of Technology

Associate Professor, Rafael Sacks
Faculty of Civil and Environmental
Engineering
Technion - Israel Institute of Technology

Assistant Professor, Jakob Beetz
Department of the Built Environment
Eindhoven University of Technology

David Orndorff
Vice President of Engineering
The Shockey Precast Group

Associate Professor Russell Gentry
School of Architecture
Georgia Institute of Technology

Date Approved: January 07, 2015

To my parents, Mohammad Bagher and Sarah

ACKNOWLEDGEMENTS

Successful completion of my PhD dissertation wouldn't have been possible without the guidance of my advisor and committee members and support of my family and friends.

My journey throughout my PhD studies has been long and at times challenging but a rewarding and exciting one. I have had the honor of working under the supervision of Professor Chuck Eastman. I am deeply gratified by his intellectual contribution, continued support and inspiring leadership. I am so thankful for the trust he placed in me that provided the opportunity for me to work as part of his research team. Professor Eastman has an exceptional experience and profound knowledge in the BIM research domain. Participating in his classes and working with him on four different research projects and authoring several papers with him have been very influential in shaping my research direction and methods. I am also very grateful for his understanding, flexibility and patience in providing me with the freedom to find and pursue my own research path. I would like to thank Professor Rafael Sacks for his thoughtful guidance and valuable advice during my PhD research work. He has been a great inspiration to me. I am extremely grateful to him for providing the opportunity of working with his research team in Technion that enabled me to implement the developed framework for the research work presented in this dissertation. I would like to thank Professor Jakob Beetz for his insightful feedback and helpful recommendations. He has been influential in initiating the direction of this research work for which I am grateful. Many thanks to David Orndorff who generously and patiently supported me during the industry knowledge acquisition of this research work. He has an excellent expertise in the precast concrete domain. His collaboration was instrumental to my

understanding of the domain and industry best practices. I would like to thank Professor Russell Gentry for reviewing my dissertation and providing feedback during my defense.

I was fortunate to work with Michael Belsky, a PhD candidate in Technion's Virtual Construction Laboratory, during the development and implementation of the rule set libraries. He provided a close collaboration on the research over the course of several months and enabled me to implement the developed framework using the SEEBIM reasoning engine that he had developed. I am very thankful to him for his valuable help and contribution and his patience and understanding. I will miss our discussions and collaborative work.

I am grateful to Professor Fried Augenbroe for his great insight and valuable guidance and his trust in me that helped me find the support I needed for continuing my research work at Georgia Tech. I would like to thank Phil Bernstein and Professor Arto Kiviniemi for their support of my research work and the stimulating discussions we had. Many thanks extended to William David Bone a dear friend and a mentor who supported me during difficult times.

I would like to express my gratitude to many construction companies that supported my research. This work would not have been successful without their generosity in sharing their expertise and vast experiences. Special thanks are extended to Bill Farnsworth from Tindall Corp., Kip Varner and Matthew Cooper from Shockey Precast, Larbi Sennour, Paul Arthur, Van Diep, and Michael King from Consulting Engineers Group (CEG), Jonathan Lazenby and Dan Wiggins from Gate Precast, Robert Adkins and Mo Kright from Castone Corp., and Jason Lien from EnCon Colorado for volunteering their time and sharing their valuable knowledge.

Finally I would like to use this opportunity to deeply thank my loving parents Mohammad Bagher, the university professor, and Sarah, the high school teacher. From my childhood they created an environment in the family in which education and learning had the

highest value and priority. They taught me to dream big and to be persistent and resilient in pursuing higher education and cherished my endeavors. Their love, unmatched care and wise guidance provided me with the strength, courage and direction to navigate this challenging journey.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iv
LIST OF FIGURES	xii
LIST OF TABLES	xvi
SUMMARY	xviii
I RESEARCH MOTIVATION AND GOALS	1
1.1 INTRODUCTION	1
1.2 PROBLEM FORMULATION	2
1.3 RESEARCH QUESTIONS.....	6
1.4 RESEARCH OBJECTIVES AND GOALS.....	8
1.5 RESEARCH IMPACT AND POSSIBLE EXTENSION	10
II LITERATURE REVIEW	13
2.1 COST ESTIMATION	13
2.1.1 COST ESTIMATION METHODS	14
2.1.2 INTUITIVE AND ANALOGICAL METHODS: EARLY DESIGN STAGES	16
2.1.3 LIMITATIONS OF INTUITIVE AND ANALOGICAL METHODS	19
2.1.4 ANALYTICAL METHODS: LATE DESIGN STAGES	20
2.1.5 ANALYSIS CONCLUSION AND ADOPTION	24
2.2 KNOWLEDGE-BASED SYSTEMS	25
2.2.1 KNOWLEDGE-BASED SYSTEMS FOR COST ESTIMATING.....	27
2.2.2 COMPONENTS OF KNOWLEDGE BASED SYSTEMS	30
III RESEARCH METHODOLOGY AND SOLUTION DEVELOPMENT	33
3.1 RESEARCH METHODS AND PROBLEM SOLVING APPROACH.....	33

3.2.	KBS FRAMEWORK DEVELOPMENT FOR PRECONSTRUCTION ACTIVITIES	38
3.2.1	MODULARIZED STRUCTURE.....	38
3.2.2	DOMAIN LAYER: KNOWLEDGE BASE.....	39
3.2.3	KNOWLEDGE ACQUISITION: METHODS AND SOURCES	41
3.3	REASONING LAYER: RULE LIBRARY AND INFERENCE ENGINE	44
3.4	VALIDATION	47
IV	SEMANTIC ENRICHMENT OF DESIGN MODELS: PRECAST	
	CONCRETE COLUMN CASE.....	50
4.1	INTRODUCTION	50
4.2.	MULTI-PARTY COLLABORATION AND SEMANTIC INTEROPERABILITY	52
4.2.1	MODEL QUERY SOLUTIONS	53
4.2.2	SEMANTIC ENRICHMENT OF MODELS	54
4.3	PROBLEM DEFINITION.....	55
4.4	PROPOSED SOLUTION FRAMEWORK.....	56
4.5	PRODUCT MODEL: PRECAST CONCRETE COLUMN	58
4.5.1	PROBLEM SOLVING METHODS AND KNOWLEDGE ROLES	60
4.6	DEVELOPMENT OF A PROBLEM SOLVING ALGORITHM.....	62
4.7	RULE SET DEVELOPMENT AND SEMANTIC MODEL ENRICHMENT FOR	
	COLUMN SEGMENTATION	65
4.7.1	SEEBIM ADOPTION.....	65
4.7.2	RULE STRUCTURE.....	66
4.7.3	RULE FUNCTION CATEGORIZATION.....	69
4.7.4	GEOMETRIC AND NON-GEOMETRIC ATTRIBUTES EXTRACTION	

AND DISCOVERY.....	70
4.7.5 FUNCTIONAL CATEGORIZATION OF OPERATORS	72
4.8 EXAMPLE RULE SETS: STRUCTURE ANALYSIS AND RESULTS	76
4.8.1 OBJECT CLASSIFICATION RULES	76
4.8.1.1 BEAM CLASSIFICATION	76
4.8.1.2 INTERNAL AND EXTERNAL COLUMN CLASSIFICATION.....	77
4.8.1.3 POCKETED AND NON-POCKETED COLUMN CLASSIFICATION.....	82
4.8.2 RULES TO FIND CLOSEST OBJECTS TO ANOTHER OBJECT IN A SPECIFIED DIRECTION	83
4.9 THE ENRICHED IFC FILE RESULTS AND FINAL PHASE.....	87
4.10 CONCLUSION AND NEXT STEPS	91
V A KNOWLEDGE BASED SYSTEM FRAMEWORK FOR AUTOMATIC EVALUATION AND PREPARATION OF BIM-BASED DESIGN FOR CONSTRUCTION	93
5.1 INTRODUCTION	93
5.2 KNOWLEDGE-BASED SYSTEMS OVERVIEW	94
5.2.1 KNOWLEDGE BASED SYSTEMS ARCHITECTURE	95
5.2.2 PROPOSED KBS FRAMEWORK FOR PRECONSTRUCTION ACTIVITIES.....	96
5.2.3 COST ESTIMATION METHODS: ADOPTION IN THE FRAMEWORK.....	98
5.3 PROBLEM DEFINITION.....	100
5.4 THE KBS FRAMEWORK IMPLEMENTATION: SOLUTION OVERVIEW.....	102
5.5 SEMANTIC ENRICHMENT OF DESIGN MODELS	104
5.5.1 DEVELOPED RULE SET FOR THE SLAB MODULARIZATION.....	104

5.6	AUTOMATIC DESIGN OF SLAB PIECES	113
5.6.1	STRUCTURAL ANALYSIS TO FIND THE MAXIMUM STRUCTURALLY FEASIBLE WIDTH OF SLABS	113
5.6.2	AUTOMATION AND OPTIMIZATION OF SLAB PIECE DESIGN	116
5.6.2.1	USER INPUT: COMPANY PREFERENCES AND LIMITATIONS.....	117
5.7	TEST CASE RESULTS	122
5.8	CONCLUSIONS	125
VI	AUTOMATED DETAILED DESIGN FOR STREAMLINED APPLICATION OF BIM IN PRECONSTRUCTION ACTIVITIES	126
6.1	INTRODUCTION	126
6.2	DESIGN AUTOMATION.....	126
6.3	INTEGRATION WITH DESIGN OPTIMIZATION METHODS	127
6.4	INTEGRATION WITH KNOWLEDGE-BASED SYSTEMS AND OBJECT-ORIENTED MODELING.....	129
6.5	AUTOMATED DETAILED DESIGN: PRECAST CONCRETE CONNECTIONS	131
6.5.1	INTRODUCTION TO PRECAST CONCRETE CONNECTIONS	134
6.6	PROTOTYPICAL IMPLEMENTATION OF THE PROPOSED SOLUTION	134
6.6.1	COLUMN TO COLUMN CONNECTIONS	136
6.6.2	BEAM TO COLUMN CONNECTIONS.....	138
6.6.3	SPANDREL TO COLUMN CONNECTIONS	138
6.6.3.1	IMPACT OF SPANDREL DESIGN CONDITIONS ON PREDICTIVE DETAILED DESIGN	140
6.6.3.2	RULE SET DEVELOPMENT FOR PREDICTIVE DETAILED DESIGN	

OF SPANDRELS	147
6.6.4 DOUBLE-TEE, SHEAR WALL AND BEAM CONNECTIONS	156
VII LIMITATIONS AND GENERALIZATION OF THE PROPOSED	
FRAMEWORK.....	159
7.1 RESEARCH LIMITATIONS.....	159
7.2 SYSTEM GENERALIZATION.....	160
7.3 SYSTEM EXTENDIBILITY TO OTHER DOMAINS	162
VIII BROADER IMPACTS AND CONCLUSIONS.....	164
APPENDIX A - RULE SET FOR AUTOMATIC SEGMENTATION OF	
PRECAST CONCRETE COLUMNS.....	168
APPENDIX B - RULE SET FOR AUTOMATIC MODULORIZATION	
OF PRECAST CONCRETE SLAB	173
APPENDIX C - RULE SET FOR AUTOMATIC DESIGN OF COLUMN	
TO COLUMN AND COLUMN TO BEAM CONNECTIONS.....	177
APPENDIX D - RULE SET FOR AUTOMATIC CLASSIFICATION AND	
DESIGN OF SPANDREL TO COLUMN CONNECTIONS	181
APPENDIX E - RULE SET FOR DESIGN OF CONNECTIONS BETWEEN	
DOUBLE-TEE SLABS, SHEAR WALLS AND BEAMS.....	193
REFERERENCES.....	204

LIST OF FIGURES

Figure 1.1: Design progress and design data availability for preconstruction activities....	5
Figure 2.1: Integrated analytical cost estimation process.....	21
Figure 2.2: Shift in the routine versus creative design time using KBE.....	27
Figure 3.1: Problem decomposition – A precast concrete beam feature- and function-mode.....	37
Figure 3.2: Developed framework for knowledge-based quantity takeoff and cost estimation.....	40
Figure 4.1: A precast concrete column product decomposition and information flow model.....	59
Figure 4.2: Example of a problem-solving method structure: inputs, outputs and actions	61
Figure 4.3: The algorithm developed for precast column segmentation.....	63
Figure 4.4: Relative spandrel-column positions; (a) outboard spandrel & pocketed column; (b) outboard spandrel but no pockets in the column.....	64
Figure 4.5: List of attributes and operators used in the column segmentation rule set....	68
Figure 4.6: column length evaluation and split pieces creation rule.....	69
Figure 4.7: Color code legend for rules.....	70
Figure 4.8: Implementation of two objects’ adjacency relationship analysis operator....	73
Figure 4.9: Range of width, height and aspect ratio for different types of precast concrete beams.....	77
Figure 4.10: Rules to identify spandrels and non-spandrel beams.....	78
Figure 4.11: Internal column classification rule.....	79

Figure 4.12: Classification of columns to be segmented like internal columns.....	80
Figure 4.13: External column classification.....	82
Figure 4.14: The enriched IFC model for column classifications.....	83
Figure 4.15: Rule I - Closest floor to internal and segmented like internal columns' centroid.....	84
Figure 4.16: Rule II - Closest floor to internal and segmented like internal columns' centroid.....	86
Figure 4.17: The enriched IFC model depicting closest floor/spandrel to columns as well as beam classification example results.....	88
Figure 4.18: Collection of snapshots from an enriched IFC P21 file created by execution of column segmentation rule sets.....	89
Figure 5.1: Knowledge based systems structure.....	96
Figure 5.2: The proposed KBS framework.....	99
Figure 5.3: Implementation of the KBS framework for precast concrete slab segmentation and quantity take-off.....	103
Figure 5.4: List and categorization of object attributes and operators used in semantic enrichment of models for slab modularization.....	105
Figure 1.5: The algorithm developed for developing the rule set required for semantic enrichment for slab modularization.....	106
Figure 5.6: Height range of different types of precast concrete slabs used for slab classification.....	107
Figure 5.7: Relationships created between model objects for identifying column bays and assigning them to slabs.....	108

Figure 5.8: The rule to build column_bay relationships.....	109
Figure 5.9: The rule to add bay lengths to column_bay relationships.....	111
Figure 5.10: Collection of snapshots from an enriched IFC P21 file created by execution of the slab modularization rule set.....	112
Figure 5.11: Added slab entities to the enriched IFC model and their attribute information	124
Figure 6.1: Current versus proposed QTO and CE process.....	132
Figure 6.2: List of attributes and operators used in the predictive design of precast concrete element connections.....	135
Figure 6.3: Load-bearing and non-load bearing spandrels: (a) LB spandrel with added corbels to support transfer of loads; (b) NLB outboard spandrel; (c) NLB inboard spandrel dapped to allow the intersecting beam's access to the column surface.....	142
Figure 6.4: Approximate load eccentricity in outboard and inboard LB spandrels.....	144
Figure 6.5: Spandrel intersecting a non-pocketed column with a notch to hide the HSS tube bracket.....	145
Figure 6.6: Various possible design situations for spandrels at the building corner.....	148
Figure 6.7: Various possible design situations for spandrels on the building edge.....	149
Figure 6.8: The rule designed to identify spandrel and slabs that are aligned in both sides	153
Figure 6.9: The rule designed to identify columns and slabs that are aligned only in one sides.....	153

Figure 6.10: The rule designed to identify and create
outboard_spandrel_pocketed_column connection relationships and
to create the required connections.....155

Figure 6.11: Design situations for Double-tee, shear wall and beam connections
(model courtesy of The Consulting Engineering Group company).....157

Figure 6.12: The enriched IFC model imported to Navisworks Manage that depicts
the added double-tee, shear wall and beam connections.....158

LIST OF TABLES

Table 4.1: comparative list of attributes and operators in SEEBIM used in this research work.....	67
Table 5.1: User input for precast concrete double-tee slab structural analysis to find max. DT width and stem reinforcement design in various loading and span conditions	113
Table 5.2: Results of the slab structural analysis including the max structurally feasible DT width and stem reinforcement design for each DT width.....	115
Table 5.3: Results of the design model semantic enrichment and performed structural analysis used as input for automated and optimized slab design....	119
Table 5.4: User input reflecting company preference and limitations used as input for automated and optimized slab design.....	119
Table 5.5: Semantically enriched IFC test model data extracted to be used as input for automated and optimized slab design.....	119
Table 5.6: Output of the automatic design of the slab pieces: slab piece lengths and widths in each floor level, total quantity of slabs in each level and size, and slab stem reinforcement design.....	123
Table 5.7: Output of the automatic design of the slab pieces: slab piece quantity and stem reinforcement in each size group in the whole project.....	124
Table 6.1: The guideline developed for predictive design of beam to column connections	139
Table 6.2: Various object relationships analyzed in developing the spandrel identification and connection design rule sets for corner spandrels.....	150

Table 6.3: Various object relationships analyzed in developing the spandrel

identification and connection design rule sets for corner spandrels.....151

SUMMARY

The current industry practices in many preconstruction and construction activities, especially quantity take off (QTO) and cost estimation (CE) activities which were closely studied in this research, remain to a large extent manual, error-prone and time-intensive, mostly relying on 2D drawings. Adoption of a BIM integrated workflow for preconstruction activities will provide the rich information embedded in parametric models to incorporate in the process, potentially enhancing the accuracy of the results. The intelligent behavior of the parametric models can automate most of the process enhancing efficiency of these processes. There are significant obstacles for providing a streamlined, efficient and practical work flow integrating BIM-assisted design information into these preconstruction activities. These obstacles have prohibited the wide adoption of BIM in these areas. Two main challenges for such a streamlined information flow throughout the AEC projects that haven't been sufficiently addressed by previous research efforts include lack of semantic interoperability and a large gap and misalignment of information between available BIM information produced by design activities and the required information for performing preconstruction and construction activities. This research effort proposes a knowledge-based system (KBS) that encapsulates domain experts' knowledge and represents it through modularized rule libraries. The goal is to first semantically enrich design models and embed the design information essential for preconstruction activities. The enriched design models are then used for automated detailed design to evaluate and classify the design objects and modify representation of the objects to demonstrate appropriate constructible product units. Subsequently, the product features and their attributes that are normally missing from the design models like connections and reinforcement elements are inferred and automatically

added to the enriched and modularized models. This research work is intended to improve accuracy and cost-effectiveness of adopting BIM in preconstruction and construction activities with a focus on QTO and CE, by providing an enriched model of a project that incorporates the expertise of domain experts. The proposed framework will assist automation of the repeated and time-consuming tasks in preconstruction and detailed structural design, enabling experts to focus more on creative aspects of these activities. It will facilitate a paradigm shift in knowledge availability in projects, disseminating construction and detailed structural engineering knowledge to designers and other parties involved in the AEC projects.

CHAPTER I

RESEARCH MOTIVATION AND GOALS

1.1 Introduction

Efficient and accurate quantity take off (QTO) and cost estimation (CE) are pivotal to a project's success. They are knowledge-intensive [1]; they are the prerequisites to many other activities in a project from budgeting, bidding and contracting to value based design, production planning and budget control; they require extracting information based on the knowledge of domain experts about the processes and their constraints throughout the lifecycle of products and projects.

A study by Sacks & Barak [2] measured the potential productivity improvement in design and detailing of building structures due to using 3D parametric modeling instead of 2D drawing. The study showed considerable productivity gain in quantity take off activities. Another study [3] explored various benefits of using BIM in the precast concrete industry reported a measured productivity improvement of 82-84% in developing detailed engineering drafts of precast concrete designs.

A study by Aram et al. [4] identified areas of potential contribution by BIM platforms in the concrete reinforcement supply chain in four categories of design and modeling, editing and updating, interoperability, and project and construction management. Requirements of BIM platforms to improve the industry performance in these four areas were identified based on a developed information process model.

Examples of assessed quantitative and qualitative enhancement in the reinforced concrete projects using BIM for budgeting and estimating were provided. A report that aggregated 26 project case studies [5] stated that the four test case projects that had used model based quantity take off experienced 25% reduction in resource investment and improved accuracy. In the one project that 3D models were linked to a cost estimating database 80% time saving was realized.

All these studies illustrate the broad benefits that preconstruction activities can expect by adopting a BIM-based process. Yet, there are significant obstacles for providing a streamlined, efficient and practical work flow integrating BIM into preconstruction activities that have prohibited wide adoption of BIM in these areas and have kept them largely manual and 2D drawing dependent.

1.2 Problem Formulation

There are commercial software products available that attempt to semi-automate these tasks through augmenting the quantitative information elicited from design models, creating pre-structured yet customizable cost databases and reducing repetitive aspects of these tasks [6].

Based on our study, QTO software products need to maintain three conditions for their successful performance (i) architectural and structural design models to be readily suitable for quantity take off and cost estimation; (ii) all the needed information to be quantitative in nature; (iii) designers' models to contain complete information needed for these tasks. In practice these conditions are rarely met. The focus here is not on users'

modeling practices and their use of correct modeling methods. Yet even when designs are correctly modeled:

1. Categories of contained information in models developed by designers and constructors and the way the information items are modeled and represented are different, as these models serve different purposes. Two examples are Cast-In-Place (CIP) and precast reinforced concrete products where the units of quantity take off and cost estimation are each concrete placement breaks and a product piece, respectively. However, the units of fabrication or casting often are not distinguished in models, which means for instance in the case of precast concrete products, elements like columns, slabs and wall panels are modeled as monolithic objects and not as column, slab or wall panel pieces. This difference leads to rework and often for preconstruction purposes, different construction parties have to create their own models from scratch.

2. The main focus of these solutions are eliciting and enhancing a set of standard quantities like volume, surface area, etc. for different products. The problem is that (a) each product type needs elicitation of a specific set of design properties for QTO and CE which can only be determined based on that product's supply chain, (b) sometimes the properties that impact cost of a product are not inherently quantitative. Current systems either don't elicit information about these properties from design models or they are represented as raw data and can't provide the user with the insight needed for decision-making. An example is product shape. Different shape parameters that impact the cost and in what value ranges their cost relationships and behavior change should be identified.

3. Amount of detailed information provided in design models before contractual agreements is different based on the adopted project delivery method. Yet, most often detailed design with complete information for rigorous cost estimation are developed late in the project lifecycle and usually for fabrication and production of products. For instance due to high time and cost required, many features of reinforced concrete products like connections that are important for accurate cost estimation of reinforced concrete products are often designed and modeled after the companies are contractually bound to the project.

This is the case in most projects including the projects that use traditional Design Bid Build delivery method. Some alternative project delivery methods like Design Build try to shift the involvement of construction entities to earlier stages of a project lifecycle, which requires detailed design information to be available for accurate cost estimation to mitigate the risks for construction entities at a time when most of this information doesn't exist [7].

Hence, and as demonstrated in Figure 1, currently QTO and CE experts mostly rely on their judgment and rules of thumb which are developed based on historical information. For unusual design situations, they seek the expertise of structural designers, plant managers, erectors and others on a case by case basis. This process is manual, time consuming and error prone.

These issues create considerable technical drawbacks for efficient and accurate model-based quantity take off and cost estimation. My field studies have shown that currently the QTO and CE processes employed by most construction subcontractors, where a detailed QTO and CE is required:

- are generally based on 2D drawings rather than 3D parametric models, as the object representation in models is not suitable for QTO and CE and design models don't include the level of detail required.
- mostly rely on the judgment of estimating experts and rules of thumb which are developed based on experience of estimators and historical data. For unusual situations estimators seek expertise of structural designers, plant managers, erectors and others on a case by case basis.
- as a result of the above two, are manual, time-consuming and error prone. Providing more accurate QTO and CE reports means allocating more resources to the tasks and having a more costly estimation process [7]. Adopting such a costly process is risky as construction companies on average win a small percent of the projects they bid on. Hence, often impact of many of the design conditions and features on cost of a project are not incorporated in the estimation.

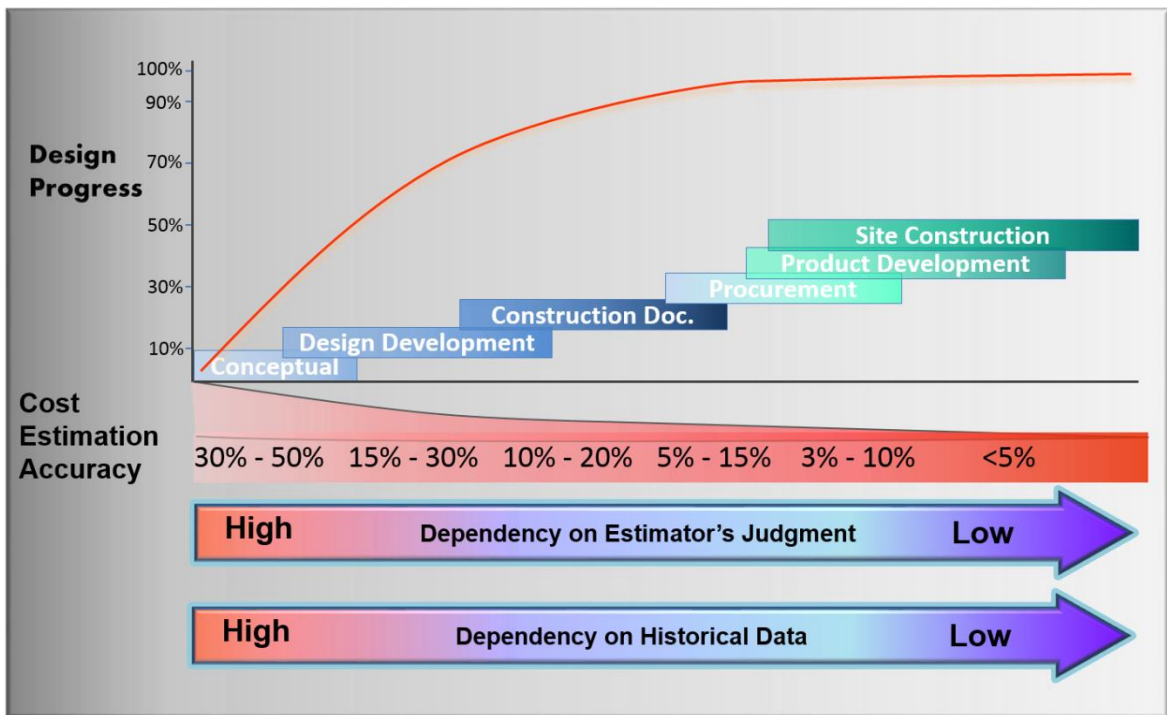


Figure 1.1: Design progress and design data availability for preconstruction activities

These difficulties are reflected in the low adoption rate of BIM based cost estimation in the AEC industry. Based on a McGraw Hill study in 2012 [8] frequency of using BIM for quantity take off and cost estimating activities is low among BIM users of all engagement levels. Three quarters of contractors with low BIM engagement level, 31% of respondents, never use it and even contractors with very high BIM engagement level have a low frequency index of 2.2 for using BIM in quantity take off and cost estimating. For 53% of non-BIM users, important factors that can influence their BIM adoption include improved budgeting and cost estimating capabilities of BIM solutions.

It is critical to rectify shortcomings of BIM platforms in providing efficient and semi-automatic QTO and CE workflows and that such improvements will promote overall BIM adoption in the AEC industry thereby providing far-reaching value to the industry that goes beyond preconstruction activities.

1.3 Research Questions

The broad question that this research work attempts to answer is:

How can 3D parametric design models be used for preconstruction activities, more specifically for quantity take off and cost estimation, in realistic business environments where considerable amount of information critical for success of those tasks is not available until late phases of a project? How can BIM integrated work flow for preconstruction be

designed to perform effectively yet without requiring the manual rebuilding of design models for domain-specific purposes?

To answer this question, this research is primarily concerned with the nature and representation of information required for BIM-enabled construction work compared to what is available in BIM-enabled design and how this gap can be filled in an efficient and automatic or semi-automatic manner.

The question above leads to several more specific questions below:

1. What are the differences between design and construction information items? How can they be identified, defined and represented in a BIM-enabled design process?
2. How can knowledge of construction experts be extracted, captured, and retrieved in earlier project stages? Can we devise a set of rules to methodically encapsulate, represent and reuse construction experts' knowledge?
3. How should the various sets of constraints including production/construction feasibility, structural design and economical optimization constraints be formulated and applied to model information?
4. What is the system framework that enables semantic enhancement of design information? How can information extracted from three sources of design models, expert knowledge and user preferences and limitations

be applied to infer new knowledge, forecast the critical construction information absent from design and provide this information to users?

5. How can the intent and results of knowledge inference and design semantic enhancement be effectively communicated with users?

1.4 Research Objectives and Goals

To enable automatic and cost-effective deployment of BIM designs for construction activities, mainly quantity take off and cost estimation as well as value-based design, by developing a knowledge-based system that facilitates automatic semantic enhancement of information extracted from design models to make their information suitable and adequate for these preconstruction activities.

In an attempt to overcome these limitations, a framework is developed for a Knowledge-Based System (KBS) to identify, define and retrieve the minimum set of model information required for quantity take off and cost estimation of building systems. The example building system selected to implement a proof of concept is precast concrete. However, the developed methodology and structure of this framework have been defined to address broader applications and is adaptable to other building systems.

This framework is designed in a way that it addresses the three above mentioned shortcomings. I have been studying and developing rule sets to enhance and represent information provided by BIM platforms in a compatible form with QTO and CE purposes. The specific set of design features and their properties, both qualitative and quantitative,

that impact the cost of a project are identified. The criteria to categorize and represent these features in groups are defined, based on parameters and their value ranges where their cost relationships change. Knowledge of domain experts is elicited and codified to forecast the properties of design features required for QTO and CE tasks but absent from design models (e.g. connections) with acceptable accuracy. The complete method will provide estimators with a complete set of design-related information required to perform a model-based cost estimation in an efficient and semi-automated way.

It is important to note that developing cost relationship formulas are different based on local economic situations, adopted supply chain technology, and resource and work breakdown structure used by different companies. Hence, developing cost relationship formulas and providing cost of a project is out of the scope of this research project. The focus here is to provide a detailed level of input for estimators earlier in the project lifecycle to use for a more accurate cost estimation and to provide this input in a cost-effective way.

The dilemma for managers is that many times they have to choose between incurring losses due to less accurate QTO and CE, and higher initial investment in more detailed and accurate QTO and CE and risking loss of the investment in case of not winning the contract [7]. Yet current obstacles to use BIM technologies for QTO and CE and automating the process makes it very costly to achieve higher level of detail in their estimation efforts. Based on the interviews with several construction companies, many companies, especially subcontractors with fewer resources, can't afford a highly detailed QTO and CE. In such an environment, a solution to automate the QTO and CE activities and replace the manual process that uses drawings with a BIM-integrated one, will enable

construction companies to achieve more detailed and accurate estimations at much lower cost.

1.5 Research Impact and Possible Extension

- *Automation and improved cost-effectiveness and accuracy of preconstruction activities.* Semantically enriched models will be able to meet many of the requirements for preconstruction activities from QTO and CE to fabrication and construction planning. The system will to a large degree eliminate the need to make a new model with all the design details required for preconstruction purposes. While there will always be unusual designs that will require manual involvement of users to adjust a model, the system can fulfill semantic enhancements for standard design situations and construction companies can shift the focus of their preconstruction human resources to more detailed estimation, detailed design optimization and creative aspects of designs.

Currently, due to large amount of preconstruction work and lack of using the computing power of BIM platforms in these tasks, many times the rules of thumb used for QTO and CE over simplify design conditions, not contributing many features and conditions that impact the cost of a project leading to less accurate estimations. Adding the computing power of BIM platforms and automating the process will enable providing a more detailed and accurate QTO and CE in a cost-effective way.

- *Conceptualization and reuse of knowledge.* The process of working with industry experts to define the rules many times involved “rule discovery”, “thought process discovery” and “reasoning reform”. Many times the thought process and

rationale behind QTO, CE and structural design decisions were not clear or were not structured. So the work involved various stages of discovery, conceptualization, formalization and sometimes modification of thought process and rules. Creating a repository that houses classified and hierarchically structured rules and allows communicating the rules and factors impacting them with users will increase transparency of preconstruction decisions both inter- and intra-organizationally. It will provide the opportunity to more efficiently customize and seamlessly share the experts' knowledge among business partners. This transparency will help standardize preconstruction practices in firms and facilitate reusing the encapsulated knowledge in different projects in a consistent manner.

Potential broader impacts are explained in the conclusion chapter.

Possible future extensions. Two major extensions to the current work include

- (i) Providing the capability of geometry creation and manipulation to reflect advice of the system on detailed design automation. Currently, while we create logical objects and provide various geometric attributes for those objects like dimensions and volume which are required as an input for QTO and CE activities, physical geometry is not created.
- (ii) Linking the developed KBS to various analysis tools that can augment or optimize the predicted design. For example, the max feasible width of double tees from the structural standpoint depends on the loading conditions and span of the double tee and requires analyzing the total stress, deflection and ultimate strength of the slab. In the current work we performed the required analyses for a range of possible conditions and developed a table. In the future by linking the KBS system to a

structural analysis platform, the data from design models and user inputs can be pushed to the right tool and output can be pulled and used as an input for the slab modularization. Then all the new design conditions can be covered.

CHAPTER II

LITERATURE REVIEW

The three pillars of this research work and thus main areas of investigation include:

- Cost estimation methods: as the target application area of the system. Hence it is important to explore different means and methods used for cost estimation and design the system to provide the design input needed by the selected cost estimation method.
- Design automation: the end goal of this work with semantic enhancement of design as a middle goal and a design automation facilitator. To be achieved by the developed rule-based KBS.
- Knowledge-based systems: As a framework of choice for this effort to enable automation of BIM-based QTO and CE and improve the accuracy and cost-effectiveness of these tasks.

2.1 Cost Estimation

Efficiency, flexibility and accuracy of cost estimation methods significantly impacts every project, product development, and corporate success. Cost estimation is performed throughout a project and product development lifecycle and according to AACE International [9] can be categorized in five classes: concept screening, feasibility study, budget authorization and control, bidding/tendering, and check/control estimate. The major complexities of cost estimation are twofold: (i) the fact that at early stages of a project

when quality of cost estimation has the highest impact on the success of a project and product outcome, there is limited information available [10]; (ii) high variety of internal and external factors from design and engineering specifications to supply chain technologies and local regulations and limitations impact the total cost. Identifying all relevant factors, systematically selecting significant predictor variables, factoring them in the model, methodically defining their relationships with cost, and finally building a robust yet flexible and extendable cost model all add to the complexity of cost estimation activities [11, 91].

In this review, current cost estimation techniques used in both the AEC and manufacturing industries have been analyzed. The analysis outcome is used to select the most suitable problem decomposition methods and cost estimation techniques for cost estimation in advanced design stages of construction projects. This in turn provides a stepping stone to design a framework for detailed quantity take off and cost estimation through extracting design model data and analyzing the extracted data.

2.1.1 Cost Estimation Methods

Numerous studies have explored and implemented different cost estimation methods for generalized uses as well as specific use cases. We found many different implementations of qualitative methods used in the early design stages both in the AEC and manufacturing industries. Research efforts focused on the quantitative and analytical methods for later design stages have mostly targeted the manufacturing domain. Important reasons include the standardized production processes and higher consistency, reliability and generalizability of measurements, resource consumption, productivity rate and time and cost of each activity in a controlled manufacturing environment.

The AEC industry's progress toward more standardization is accompanied with proliferation of two major trends of prefabrication and modularization. Many trades of the AEC industry and especially prefabrication sectors such as the precast concrete industry are increasingly using analytical cost estimation methods. The controlled production environment in construction prefabrication resembles that of the manufacturing industry. Thus, the lessons drawn from manufacturing including analytical cost estimation methods can provide useful insights for implementing them in areas like precast concrete which is the main focus area of this research effort.

Researchers have categorized cost estimation techniques in a variety of ways: Cavaliere et al. [10] classified cost estimation methods as analogy-based, parametric and engineering models. Niazi et al. [11] further divided intuitive methods into Case-Based Reasoning (CBR) and decision support systems, analogical methods into regression analysis and Artificial Neural Network (ANN), and analytical methods into breakdown, operation-, tolerance-, feature-, and activity-based cost modeling. Chougule & Ravi [12] classified cost estimation methods as intuitive, analogical, analytical, feature-based and parametric.

In both construction and manufacturing industries, the amount and level of detail of available design information at each stage of a project and the purpose of cost estimation determine the feasibility and suitability of the various cost estimation methodologies. Available information and cost estimation purpose are in turn dependent upon project phase and degree of design completion. Hence, the project phase provides a good basis for categorizing cost estimation research and methods.

2.1.2 Intuitive and Analogical Methods: Early Design Stages

Numerous studies have focused on conceptual design and initial design development stages of products and projects. Due to the lack of complete design information in early stages of a project, cost estimation solutions use qualitative methods in which new projects and products are compared to previous similar ones to identify the weight of different variables and degree of similarity in important aspects of projects, which are established by the researchers. As such, they are mostly categorized as analogical decision support systems [11].

In response to limitations of traditional statistical techniques and to improve their performance in terms of accuracy and consistency, new techniques including the non-linear machine learning method of Artificial Neural Networks (ANNs), the problem-solving and learning method of Case-Based Reasoning (CBR), heuristic optimization algorithms like Genetic Algorithm (GA), and probability distribution optimization methods like Monte Carlo, and decision trees were introduced.

Two of the most frequently studied cost estimation methods for early design stages are ANN and CBR. The major advantages reported for ANN models are that they do not require the project cost to be defined as a specific function of cost-affecting variables. Also many studies in both construction and manufacturing have shown their higher accuracy compared to traditional regression models [10, 13, 14]. Major advantages of CBR models are transparency of the process which turns it into a suitable decision support tool, the ability to handle missing attribute information from previous cases and the feasibility of long-term use due to ease of updating models through the addition of new cases [10, 15].

The goal of these methods is to predict project costs with limited information provided in early stages of a project with an acceptable accuracy rate. These cost predictions are generally used by project owners for feasibility studies and budgeting purposes. While several different techniques are utilized for an early stage cost prediction, the applied methodologies are comparable in many aspects and can be generalized as the following steps:

- *Data collection* from previous sample projects of the same type and identification of important cost-driving attributes in the projects. These attributes and their values are used as inputs for the cost estimating system where the total project cost is the output. These are high level inputs. One example involves ten attributes of project type, scope, year, season, location, duration, size, capacity, water bodies, and soil condition which were used in a cost prediction study for highway projects [16]. Another study [17] collected values of 6 LEED certification categories in addition to building type, year and location data and used them as the system inputs to predict LEED certified projects' cost premium.

- *Identification and assignment of the optimal weights to input attributes* using different methods from linear statistical methods like Multiple Regression Analysis (MRA) [18, 23] to ANN [17, 19], GA [20] and decision trees [21].

In these methods usually data from part of the collected project cases is used to train the model. The rest of collected project cases are used to test and validate performance of the built model in predicting total costs of projects, using the assigned weights for different attributes. The training involves systematically adjusting weights of attributes through comparing *predicted* output of the model – here the project cost – to the *actual*

project cost. The goal is to minimize the error between predicted output of the model and the actual project output. One training method example is the back propagation algorithm which is the most broadly used method in ANNs. In this method, Mean Square Errors (MSE) are measured and minimized.

- *Prediction power assessment of the system.* Quality of a cost estimating model is evaluated by measuring its performance in predicting a project's cost using the final assigned weights for different attributes. As mentioned earlier, some of the collected project cases are used to compare predicted outputs of a model to the actual costs of those projects. Various algorithms and statistical methods can be used to assess the prediction power. For example, in the MRA method, the R^2 , the coefficient of multiple determination, or the adjusted R^2 (\bar{R}^2) is used where the closer its value to 1, the higher the model's cost prediction accuracy. In the CBR method, different algorithms like the nearest-neighbor algorithm are used to calculate the similarity of the test project to training projects by a methodic comparison of their attributes. Finally the project case with the highest similarity rate is retrieved [22].

In the CBR method, the retrieved project is revised and adapted to the test project. Some CBR studies have applied subjective model revision approaches, while a few have used a systematic and analytical revision and adaptation process; one example is a construction CBR study that has applied a MRA-based process for revision [23]. Marzouk & Ahmed [24] used four methods of null, weighted, neuro and fuzzy adaptation to revise the retrieved manufacturing cases.

2.1.3 Limitations of Intuitive and Analogical Methods

Part of the shortcomings of cost prediction methods stems from their inherent nature that inevitably rely on the availability of data from past similar projects. Methods like ANN can achieve more accurate results with fewer historical projects compared to traditional methods. Yet, they need a substantial number of similar historical projects with known project costs and cost driving attribute values [13]. This not only prompts feasibility issues due to rather scarce construction projects' data, but also hinders wide application of these methods because of the considerable time and funding needed to collect the required data. Better methods for reliable handling of incomplete historical data should be investigated [25].

A few studies have tried to apply a systematic process to attribute selection. For example, [24] conducted a statistical analysis on the results of a survey about cost driving factors in the pump station projects to identify the factors with the highest cost impact. While attributes selected for inputs of a cost estimation model significantly impact accuracy of predictions of the built models, most studies haven't analytically established that the selected attributes are the most critical cost driving factors of the selected test project. Often selected attributes were just a subset of what could be easily determined and collected from early stages of historical projects or were based on selections of previous studies.

Moreover, while the improved techniques that different cost estimation methods use to improve the accuracy of their cost prediction models most of the studies haven't explored situations where the results are not satisfactory.

Shortcomings specific to each method have been determined and analyzed in numerous studies. Important examples are the difficulty in handling large numbers of variables (project attributes) and the requirement for establishing a cost function between inputs and outputs by regression analysis methods [10]. ANN models have been reported to require considerable time and effort to retrain and update when new cases are added, making them unsuitable for long-term use. Moreover, unknown relationships of inputs and outputs in the hidden layers result in a black box technique. Providing analytical explanations for the process and results to decision makers is thus difficult [14].

Furthermore, these methods and researches have not considered cost effects of technological changes such as process automation, prefabrication and Building Information Modeling (BIM). Other issues to be investigated include alternative contract types like design-build and IPD that allow concurrent design and construction, the selected structural, production and construction methods, and unusual design forms on their analogy and outcome.

2.1.4 Analytical Methods: Late Design Stages

Methods used in late design stages attempt to analyze a product design and its supply chain processes in detail to achieve more accurate cost estimation. As such, they can be categorized as quantitative or analytical methods and can be further divided into three categories of function-, feature- and activity-based cost estimation. Boundaries between these methods are blurred, and studies sometimes use a collection of cost factors associated

with production processes, morphological design features, and consumed resources. Figure 2 summarizes the methodology used by the analytical methods. Analytical methods at use a product decomposition structure and later need to integrate the collected knowledge about features, functions and activities. The analytical methods vary in terms of level of granularity present in their models.

An activity-based parametric solution for estimating cost of the foundry stage of disk brake production was developed in a study by Qian & Ben-Arieh [26]. Major activities and their total cost of production were identified. Activities were divided into three categories: (i) activities with fixed costs in the batch level, (ii) activities with variable costs in the batch level and a linear relationship with the batch size, (iii) activities with diseconomy of scale. The major cost driver for activity i was defined (e.g. machining hour for the testing activity).

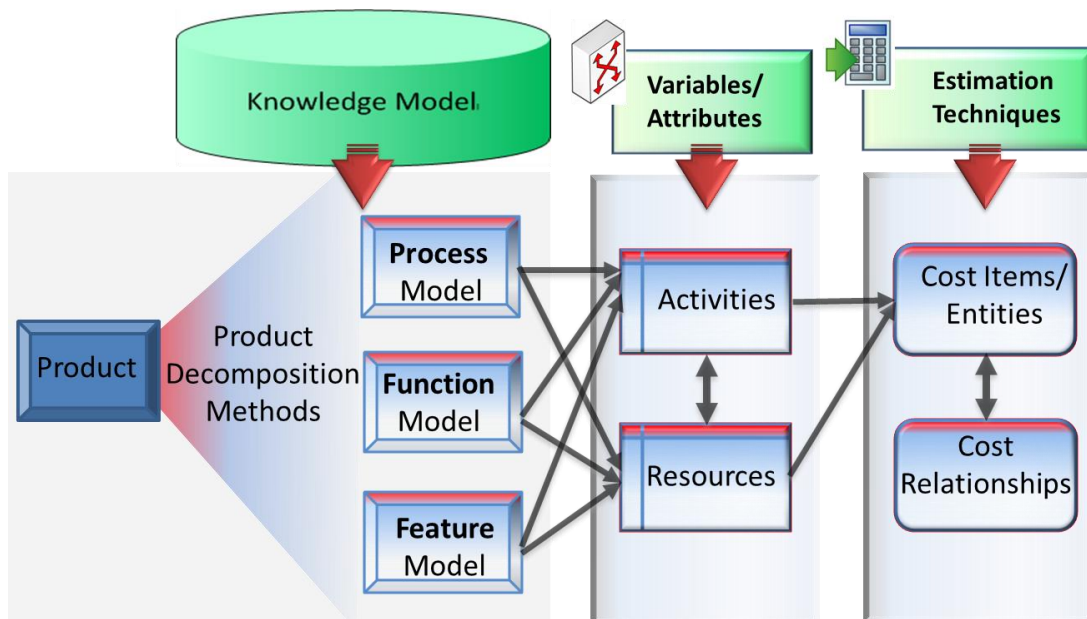


Figure 2.1: Integrated analytical cost estimation process

These parametric cost estimation studies have been mostly performed in the context of manufacturing industry and scope of studies has been typically limited to one part type and one phase of the production with limited parameters and activities.

In another study [27], the cost of manufacturing was estimated by modeling resources required for each activity and aggregating them to estimate the cost of operation process of features of the product. A product model describing the product from the manufacturing point of view was developed. The different available operations and alternate machine uses were identified for each feature. The cost reasoning model estimated the total cost as the sum of the manufacturing operations costs of all product features through solving a constraint satisfaction problem.

In the study by Roy et al. [28] to estimate cost of an automotive exhaust system production, the product was functionally decomposed, specification parameters describing each function were identified, historical data regarding processes and resource consumption rates were collected, and finally cost items were linked to each function to estimate cost of adding each function to the product.

A study by Chougule & Ravi [12] created a system in which cost of activity resources were calculated using (i) various geometry, quality and production attributes of the product; (ii) a process model; and (iii) a 3D model for feeding and gating systems, as inputs of the cost model. Another study [29] developed a mathematical model to minimize cost of the concrete structures while satisfying structural strength and stiffness requirements.

Based on the reviewed research efforts on the analytical cost estimation the following methodology can be formulated:

- *Product Decomposition.* One of the product decomposition methods is selected. A product decomposition model for the standard product design is developed. Optional functions or features and alternative processes are defined. After an initial design, parts of the designed model that are of high complexity or of cost significance should be further broken down to achieve an appropriate level of detail.

- *Data collection.* Data regarding product, process, projects, functions, and cost driving parameters are collected from various resources including historical databases, engineering specifications, recording production supply chain, expert knowledge and judgments. This data is used to identify cost driving parameters and their relationship with total cost of each activity, function or feature. Evaluating the quality of acquired data to ensure of its measurability, reliability and completeness [10] is important for defining accurate cost functions.

- *Cost driving parameters/variables/attributes are specified* for each unit of the decomposition model –i.e. each activity, function or feature –through analyzing the supply chain and eliciting knowledge of domain experts. For accurate cost estimation, selected attributes should reflect all aspects of a product’s lifecycle. Various categories of parameters concerning geometry, quality, aesthetical requirements, engineering performance and production technology should be analyzed.

- *Define cost relationships/functions.* Cost behavior of units of the product decomposition model with regard to changes in the magnitude of those units is analyzed. These cost functions are expressed mathematically by equations between

parameters defining each unit to total cost of the unit which basically requires a regression analysis. Usually an operation process involving several activities is required to produce a feature or provide a function. Hence, analyzing cost behavior of functions and features often leads to further decomposing them into activities.

In terms of cost relationships, activities most frequently belong to one of the four types of (1) fixed, (2) variable (proportionate to activity volume), (3) mixed (with a fixed and a variable cost portion), and (4) step (fixed within specific activity volume intervals, but jump to a higher level from one interval to the next) [30]. In some cases activities have nonlinear and sometimes multi-variable cost relationships. In late design stages and in presence of the complete required design data and with a sufficient level of detail in decomposing a product, the cost behavior of activities can be adequately approximated by a linear function.

- *Aggregate cost relationships and estimates.* Aggregation of cost functions for all units of product decomposition model provides the total product cost. Cost aggregation can be done on various levels and each provides a unique insight into the product cost: (i) when a variable affects several different activities or features and hence is repeated in different cost functions, these functions can be aggregated to analyze the overall impact of each variable on the total product cost; (ii) all cost functions related to each specific resource can be aggregated throughout the supply chain to identify resources that comprise the largest portion of the overall cost; (iii) aggregation of the cost of activities at each stage or sub-process to focus on stages

or processes with highest share of total costs or higher cost rates than industry averages.

2.1.5 Analysis Conclusion and Adoption

CE methods used in early stages of a project mostly can work only with a limited number of variables and provide a rough approximation of cost of a project suitable for budgeting. Considering this, they are not suitable for a more detailed CE process when there are more design information available and for instance geometry of building and different spaces within a building, type of building structure and location of structural elements are determined. Hence, for this research work, I adopted an analytical CE method. The main takeaway here was to define a method to analyze and decompose different products to their basic features, functions and processes. This decomposition provides a basis to identify the parameters that determine cost of each feature, function and process. Then sources and methods to extract and represent value of these parameters are identified. An example of a product decomposition and how it is developed will be provided in Chapter 4. Results of the study of these CE methods used in different project stages and analysis of the performance and shortcomings of each method has been published by the author in ISARC 2014 conference [31].

2.2 Knowledge-based Systems

Knowledge-Based Systems (KBS) have emerged from the Artificial Intelligence (AI) field and are employed for numerous purposes in various industries. KBS are systems that acquire, represent and process data, information and knowledge to generate new

knowledge. Unlike traditional information systems they can act as decision makers and serve like an expert on demand [54, 55].

Knowledge in the sense that is used in KBS can be defined as a system, that provides the ability to manipulate, transform or create data and information to make a decision, perform skillfully or solve a problem [56]. One useful classification of knowledge that grasps two of its important dimensions is (i) conceptual knowledge that is “understanding of the principles that govern the domain and of the interrelations between pieces of knowledge in a domain” versus procedural knowledge defined as “action sequences for solving problems” [57]; (ii) explicit knowledge that involves articulated and structured or semi-structured knowledge versus tacit knowledge built by experience, guided by intuition and residing in one’s subconscious [58].

A closely related concept to KBS is Knowledge Based Engineering (KBE). Various definitions have been provided for KBE and have usually classified it as a special type of KBS. One of the early definitions of KBE provided by Chapman & Pinfold [59] defined KBE as “an engineering method that represents a merging of object oriented programming (OOP), Artificial Intelligence (AI) techniques and computer-aided design technologies, giving benefit to customized or variant design automation solutions.”

Later Cooper & La Rocca [60] defined KBE as ‘the use of dedicated software language tools (i.e. KBE systems) in order to capture and re-use product and process engineering knowledge in a convenient and maintainable fashion. The ultimate objective

of KBE is to reduce the time and cost of product development by automating repetitive, non-creative design tasks and by supporting multidisciplinary integration in the conceptual phase of the design process and beyond.” According to a review by Verhagen et al. [61] KBE definitions have evolved from older narrow and technology-driven definitions to a wider and less restrictive ones.

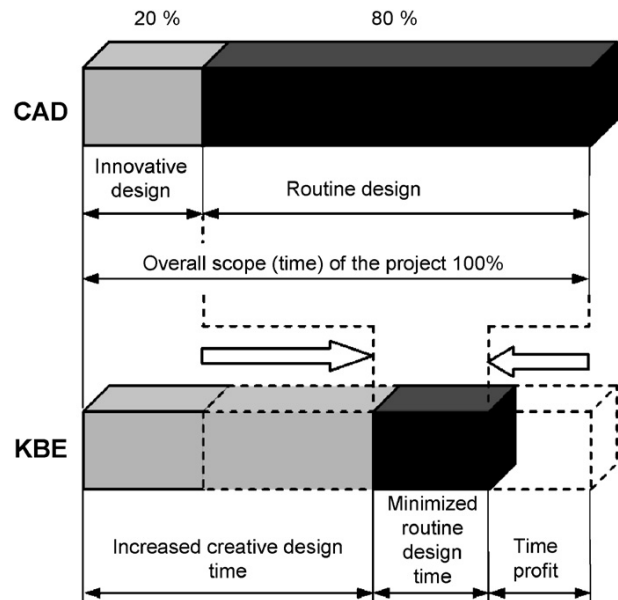


Figure 2.2: Shift in the routine versus creative design time using KBE [94]

The notion in the KBE definition of Cooper & La Rocca that identifies automation of repetitive and non-creative design tasks as one of the major benefits of implementing KBE systems, is shared by other researchers. This concept, illustrated in Figure 3 [94], highlights the fact that by significant time and cost savings resulted from automation of repetitive tasks, designers can focus more of their efforts on creative aspects of design [61].

2.2.4 Knowledge-Based Systems for Cost Estimating

Developing and using KBS for cost estimation in the manufacturing and AEC industry started in the 1990s and has continued till now with an increased interest in expanding their applications to the web. Numerous research efforts [1, 62, 63, 64, 65, 66, 67, and 68] have developed knowledge-based systems for product and project cost estimation purposes.

Some have attempted to create a framework for a broad application area but most have focused on one specific application area. These solutions have employed various cost estimation methods from intuitive and analogical to analytical and parametric [69].

Some of these systems were developed both as a decision-making support system for choosing the manufacturing process, machines and material of products and as a cost estimation solution based on the selected options. For example, Chan & Lewis [63] developed a knowledge-based system incorporating product design, process and cost knowledge into inference engines used for material and process selection and ultimately for cost estimation.

An example in the manufacturing industry is the system developed by Shehab & Abdalla [62] for modeling cost of machining components as well as molded components. The system's inputs include a material, a mold and a processing database as well as geometric and feature data of the product design model. Domain knowledge was represented in an expert system toolkit through frames and rules like material selection rules and manufacturing process and tool selection rules based on various characteristics such as material cost, product functionality and machine availability. Based on the system's recommended process, the product's manufacturing cost was estimated. While some product features like number of cavities and surface finish were factored in the estimated cost, it is not clear how qualitative aspects like shape complexity were contributed to the cost model.

Another research effort [70] acquired domain experts' knowledge about the lifecycle cost of jet engine flanges. It created a KBS that was tied to a lifecycle cost simulation model developed to analyze 3D designs based on their lifecycle cost and provide

feedback to designers. The goal was to make downstream knowledge available during early design stages.

A diverse team sponsored by the National Institute of Standards and Technology sponsored Advanced Technology Program (NIST ATP) developed the Federated Intelligent Product EnviRonment (FIPER) [66] knowledge-driven environment for concurrent engineering to reduce cost of product development. In FIPER product cost information is integrated with the knowledge base. Koonce et al. [67] developed a cost with the goal of providing an integrated web-based estimation tool in which they used the design data provided by FIPER at different stages of design completion. They integrated the design data with a cost engine consisted of Work Breakdown Structure (WBS) elements and element attributes that determine the cost of an element using a hierarchical structure for attribute inheritance.

Knowledge-based systems have been developed for various purposes for the Architecture, Engineering and Construction (AEC) industry as well. For the cost estimation domain, Staub-French et al. [1] proposed a reasoning process based on cost estimators' knowledge to represent and apply their rationale about impact of design features on cost estimation. This process customizes the activities and allocation of resources to each activity to account for project-specific features. Lee et al. [71] developed a framework that uses an ontology designed for work conditions and work items in tiling and through reasoning rules automatically selects the most appropriate work item. The inference process is designed based on knowledge of an expert and the selected work items are then used for cost estimation. In both of these efforts the focus has been on developing an

ontology to represent different design and construction conditions that affect the cost of a project.

Another research effort [72] focused on developing a production planning system for bespoke precast concrete products in which a knowledge-based solution was provided to extract geometry and other product properties from 2D design drawings using rule-based object recognition. In this solution, manual modularization of the 2D design drawings by precast fabricators, and preparation of detailed design drafts were the prerequisites for the retrieval of object properties. Another initiative [73] developed a KBS for integrating CAD systems with structural analysis and quantity and cost data. The system had an interface with available CAD software and performed a preliminary analysis of steel columns, beams and joints based on AISC and CISC code and connected the results to member cost data to generate a project quotation.

The reviewed KBSs all assume that product models used for cost estimation include all the information about feature properties that impact projects' cost and that the unit of products represented in product design models fit the cost units of manufacturers. In other words, they only extract information represented explicitly in design models, but cannot modify the design to reflect the fabrication and installation units critical for cost estimation. They do not anticipate product features missing from design until very late stages of a project nor attempt to enhance the information retrieved from design models to contribute to a project' cost estimation.

These systems would only work under ideal situations when late project information is available early in the project for design entities and is represented in design models, which is relatively rarely the case.

2.2.5 Components of Knowledge Based Systems

Domain Layer: Domain layer consists of a knowledge base which is a repository that represents the knowledge acquired from various domains and represented using different representation tools. Knowledge acquisition and representation deal with content and format of knowledge respectively and enhance availability and usability of knowledge [76]. Various textual, graphical and computer-interpretable knowledge representation conventions and tools have been developed to standardize knowledge modeling in different domains. Examples include UML and family of IDEF languages [78].

A knowledge base represents the acquired domain knowledge using an ontology. Ontologies, originally defined by Gruber [79] as “explicit specification of a conceptualization”, are fundamental for sharing and reusing knowledge. An ontology specifies a vocabulary - set of representable objects, their properties and relationships – for a universe of discourse. KBSs model their domain of interest through explicit abstraction hierarchies and rules about their relations that comprise an ontology. Shared ontologies tie modules of a KBS and are essential for communication and reuse of knowledge among different modules of one knowledge base and for integrating knowledge base of separate KBSs [75].

Reasoning Layer: The reasoning layer includes modules of rule libraries and inference engines. Reasoning processes in this layer are outlined by utilizing the concept of a Problem-Solving Method (PSM) which specifies the logics behind the reasoning processes. A PSM determines required inference actions, their dependencies and sequence as well as role of each acquired knowledge piece, namely observables, abstract

observables, solution abstractions and solutions to reach a specific goal [74]. Notion of a shared ontology facilitates implementation of a modularized structure for the reasoning layer where different modules computationally work as an integrated whole.

Task Layer: While hierarchy and relations of tasks are defined in the reasoning layer, a finer decomposing of tasks to the goal, required input, expected output and the strategy applied to generate the output is provided in the task layer [80]. Decomposing a KBS in this way allows having several hierarchies of tasks where tasks can be mixed and matched and different task compositions can be built to solve various problems.

Interface Layer: User interface systems enable interactions of KBSs with users [76]. For efficient communication, these interactions should consist of two main aspects of (a) receiving inputs from users that outline users' organization preferences, limitations or requirements. These inputs are used during the reasoning process to refine problem-solving strategies and achieve a dynamic and customized solution based on users' needs; (b) representing the outputs of reasoning and task layer based on users' criteria for selecting, filtering and grouping outputs.

CHAPTER III

RESEARCH METHODOLOGY AND SOLUTION DEVELOPMENT

3.1 Research Methods and Problem Solving Approach

The current research effort tackles the design automation problem by developing a KBS framework integrated with parametric object-based modeling schemas to automate acquisition and structuring of the design data and the domain experts' knowledge and to facilitate the reuse of acquired knowledge in broad design conditions.

The developed methodology intends to address the research questions and achieve its goals through the following methods:

- (i) *Semantic enhancement of design*: Enhancing models by transforming implicit information to explicit ones or calculating and creating new object attributes to provide all the necessary design information for QTO and CE activities. The task-essential set of information items for each object type is identified based on the defined product decomposition models. Most semantic enhancement attributes and operators as illustrated in Figure 11 are general; the attributes can be defined for and operators can be applied to a broad range of object types, and, they can be mixed and matched to create a wide variety of rules.
- (ii) *Task-based design evaluation and preparation*: As explained earlier, while the unit of QTO and CE for precast concrete products is a precast concrete piece, in the architectural design and early structural design models often pieces of precast

concrete products are not correctly distinguished. This capability automates the process of critiquing the design for manufacturability, constructability and cost performance currently performed manually by cost estimation experts. Through extracting the geometric and spatial relation information of products from models, semantically enhancing the model information and applying modularization rules developed based on the acquired domain knowledge, precast concrete model objects are properly segmented to represent acceptable approximations of precast concrete pieces that can then be used for preconstruction activities and detailed design.

This capability automates design evaluation, preparation and adjustment and eliminates the need to create new models for preconstruction purposes. Geometric and non-geometric attributes of the precast concrete pieces including dimensions, surface areas, volume and weight measurements can be derived from the existing object models and based on the predefined rules explained later in section 5.2.

(iii) *Automated detailed design*: This part involves predicting the design information about product features absent from design models. Detailed design of many key cost-driving components of precast concrete products like connections, reinforcement and form stripping and lifting inserts for the most part is performed by structural engineers who work for trade contractors. The process is costly and time consuming and normally is performed after winning the bid and securing the project and before the fabrication and construction. During the preconstruction activities information related to these components are mostly absent from models.

Similar to the model enhancing process, set of information items for each object type is identified and attribute values for each inferred from design models. Those component attributes that are important for domain tasks (e.g. number and type of reinforcement elements) are identified using the developed product models and values of those essential attributes are calculated.

The designed methodology involves the following steps:

- *Product and process studies:* Study the supply chain of both architectural and structural precast concrete products. Investigated different cost estimation conventions practiced in the precast concrete industry in the USA. Analyzed performance of the different cost estimation methods and documented the results of the study in [31]. These studies aimed to identify the weaknesses in current practices, opportunities for a BIM integrated KBS to improve and to define the goals for the research effort. An example product feature model is shown in Figure 5.
- *Problem decomposition:*
 - Devise a combined feature- and function-based analytical cost estimation method [31] as the most suitable one for the intended estimation level of detail and accuracy.
 - Decompose precast concrete products into their functional components and identified features required for each function.
 - Develop a process map for quantity take off and cost estimation for each function and feature.

- Identify cost-driving attributes of each feature and specify the parameters required to measure the impact of each attribute on cost of a project. These variables comprise the information items necessary for precast concrete cost estimation.
- Define the rule sets to infer knowledge about the required information items either typically implicit in design models or absent from models
- *Knowledge acquisition:* Identify different sources of knowledge for the domain of discourse; captured the relevant knowledge and validated the acquired knowledge. Strategize the direction and focus of the work based on the importance of each subdomain and the available opportunities to improve them. The goal of knowledge acquisition is to learn the methods and processes used by domain experts to figure out values of the parameters that compose features and activities that in turn make up a product type and affect cost of a product and ultimately a project.
- *Knowledge formalization:* Develop a rule library comprised of sets of rules to infer knowledge about the information items typically implicit in or absent from design models, specially before completion of detailed design, based on the available design information and company and project information provided by users. These rules might use different mathematical, statistical or heuristic methods to achieve the value of a parameter based on the existing information.
- *Knowledge representation:* Structure and represent the rule sets using the IFC schema as the medium to represent the structured knowledge. The IFC is the most widely accepted data standard in the AEC industry that enables the interoperability among different BIM platforms. This includes mapping the rules to the IFC data

structure including objects, object hierarchies and object properties used in the rule sets to the IFC data structure.

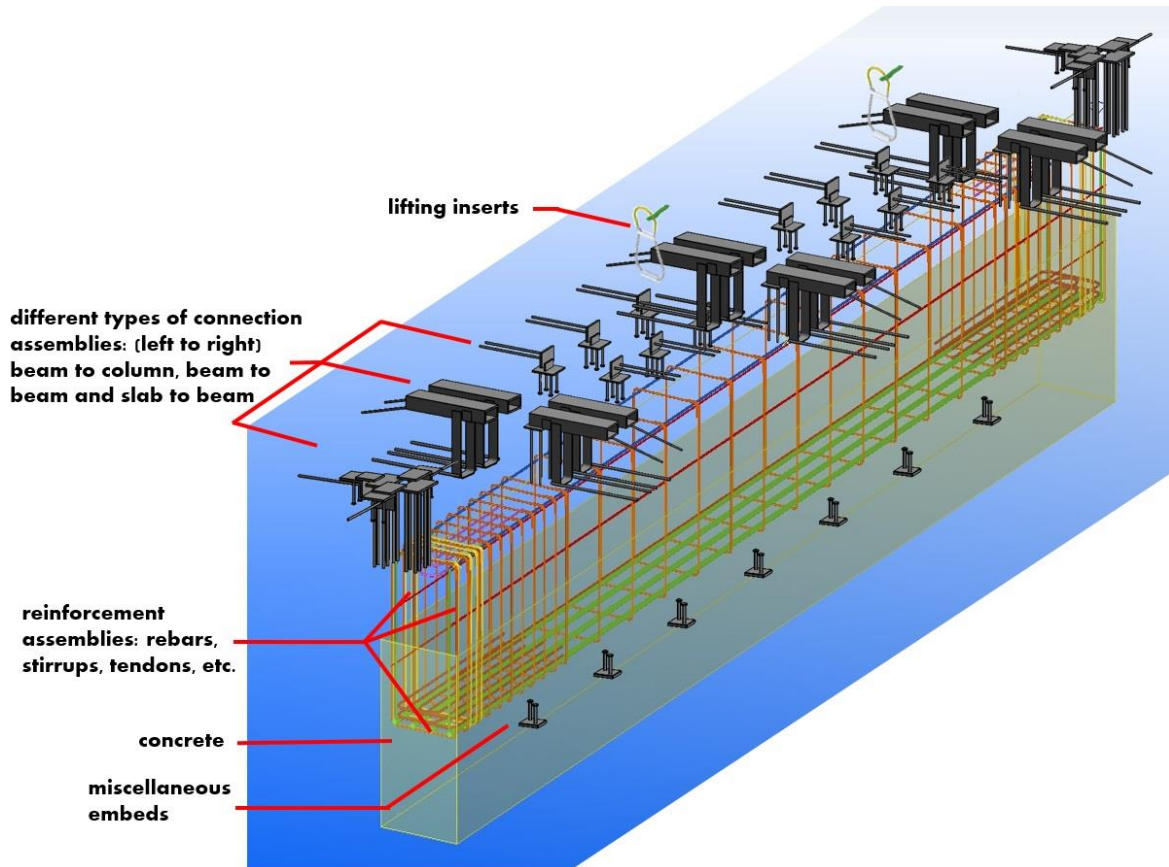


Figure 3.1: Product decomposition – A precast concrete beam feature- and function-model

- *System implementation:* Adopt the rule engine and the user interface under development for the Semantic Enrichment Engine for Building Information Modeling (SEEBIM) project [81] that uses the IFC Viewer [83] tool that reads the model information from the IFC data files. IFC Viewer is built around the IFC Engine DLL [82] that generates 3D geometry based on the IFC schema. Part of the rule engine capabilities needed to run the rule sets developed for this research work

have already been developed by the research team at the BIM lab of Technion and the rest is collaboratively being identified, defined and developed. A list of defined attributes and operators that is used in the example rule set for column segmentation is provided later in Figure 11.

3.2 KBS Framework Development for Preconstruction Activities

We have developed a KBS framework to provide a streamlined, 3D parametric model based quantity take off and cost estimation for construction products. This framework is represented in Figure 6 and includes the 4 layers of domain, reasoning, task and interface, designed for the precast concrete products which comprises the area chosen to implement a proof of concept for this research effort. This is an ongoing effort and so far the focus has been on developing a knowledge base and rule libraries.

Several precast companies have collaborated and provided their company standards, practice manuals and their historical project cost estimation information. The principal researcher of this effort co-located for a few weeks with company experts to collect information from estimators, structural engineers, plant managers and erectors; to observe their QTO and CE process; and to formulate the inference rules with the help of these experts. The knowledge base and reasoning rules are being developed both for architectural and structural precast concrete products.

3.2.1 Modularized Structure

The basis for the proposed system is modularizing the whole design into components defined by users and developing rule libraries for each module. These User Defined

Features (UDF) will provide the container needed for storing, distributing and processing and reusing the acquired knowledge [84].

The key here is to develop a data structure that identifies the parameters needed to define different categories of features, provides all-encompassing parameter definitions for various design situations and distributes them to relevant features. Yet it allows the users to share their knowledge about company practices as well as local or national industry practices by deciding which parameters they want to use, the value of the parameters and the measurement method for each parameter. As long as the user input follows the data structure defined for each product type and the general constraints, they are accepted. This provides a robust yet flexible rule development archetype.

3.2.2 Domain Layer: Knowledge Base

The domains studied in order to develop an example knowledge base that guided the listed steps from product decomposition to process mapping and rule development included architectural and structural design, and supply chain analysis (fabrication, transportation and erection) of precast concrete products. The focus of knowledge acquisition was on those domain aspects that are interdependent with quantity take off and cost.

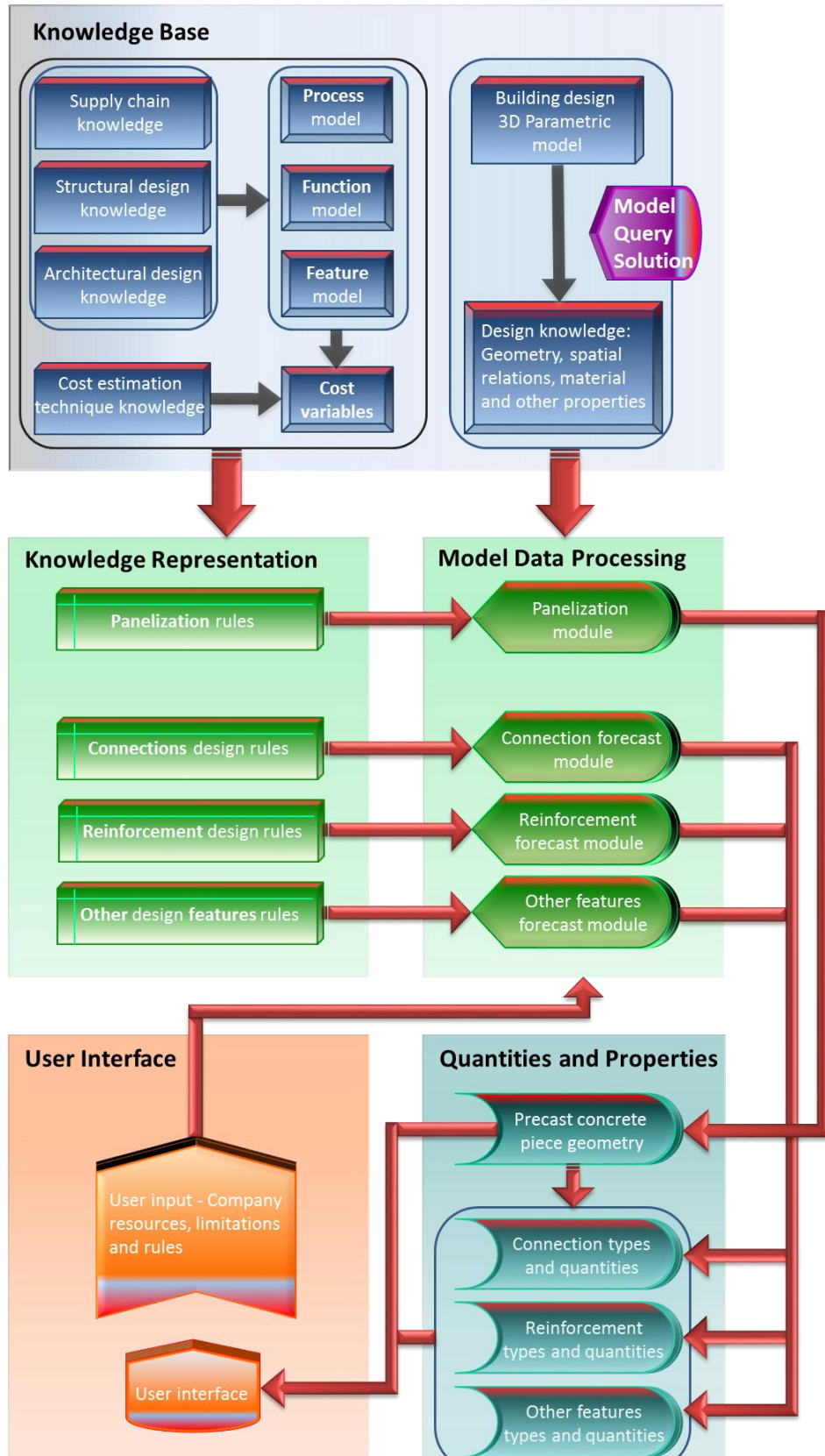


Figure 3.2: Developed framework for knowledge-based quantity takeoff and cost estimation

3.2.3 Knowledge acquisition: Methods and Sources

Knowledge acquisition sources used for this research work include:

- International and national standards: Examples of standards practiced in the precast concrete industry and referred to in this research work are the International Building Code (IBC) [85] and national design codes like those published by the Precast Concrete Institute (PCI) [86, 87, 88], by the American Concrete Institute (ACI) [89] and Architectural Precast Connection Guide published by National Precast Concrete Association (NCPA) [90].
- Historical data collected by different companies: Different companies have collected project cost information in different level of details from project cost to product cost, product feature cost and finally to materials and part cost used in each feature.
- Industrial engineering data: data collected about the time and cost of labor for each activity, cost of material and equipment under standard situations.
- Company standards: different companies over the years have developed a standard design book based on the layout of their plants, their forms, their suppliers and the type of projects they specialize that guide them throughout the design for instance for the types of connection designs in different situations and reinforcement assemblies. These standards reflect company preferences and while they sometimes develop creative designs that optimize company sources to a large degree design solutions overlap in the industry especially in the structural concrete precast that is more standardized.

- Domain experts: As the knowledge-based system used in the title of this research effort implies domain expert have been the critical source of knowledge in this work. Knowledge acquisition has been conducted by weeks of relocation to company offices and continued long distance communications with experts over the course of a year. Experts from the Beck and DPR general contractors, the Consulting Engineers Group (CEG) that specializes in precast concrete design and provides services to many precast concrete companies in the USA and overseas as well as several precast concrete companies including Tindall Corp., Shockey Precast, Gate Precast, Castone Corp., and EnCon Colorado collaborated on this project.

The specific aspects that distinguishes domain experts from other knowledge sources include:

- *Connecting industry practices to design standards:* Domain experts based on their vast experience provide valuable insight about the preferences by different companies, trends in different parts of the industry including recent changes and the future outlook, about the right interpretation of design codes for various design situations, and short cuts, customization, and implementation of design standards across the industry.
- *Insight into “why” in design decisions:* They not only share the knowledge as to “what” and “how” in their decision-making process but also as to “why” which many times standards fail to illustrate. They provide additional insight about the rationale behind their decisions and the criteria that guides their decisions to choose from feasible solutions: how project-based factors including architectural and

structural design intent, building type and size, and so on as well as external factors like the current technologies used by different companies and macroeconomic factors affect both constructability and economy of each design decision. Understanding underlying factors affecting design decision is crucial for forming quality rules that cover a wide variety of situations and strike the right balance between generalization and specialization.

- *Focus area and strategic direction:* Another aspect in decision-making that is important for formulating the rules is “frequency”: (1) How often does a specific design condition occur? (2) How often is a specific solution used? No amount of effort, in a practical manner, can cover all possible design conditions or all feasible solutions. A good knowledge acquisition strategy is to use the 80/20 rule in collecting information and forming and implementing the rules: To focus the research efforts on a subsection of the domain of discourse that accounts for a large percentage of problems and/or solutions. Only domain experts with vast experience can provide such an insight.

Two major difficulties involved with domain expert knowledge acquisition are:

- (i) *Managing tacit knowledge:* Explicit and tacit knowledge were defined in the section 2.3. Tacit knowledge comprises a large chunk of experts’ knowledge which usually deals with “why” and “how” questions. Tacit knowledge is difficult to express and transfer and is unstructured. Hence, the process of capturing the tacit knowledge and transforming it into explicit knowledge is a challenging task [93].

- (ii) *Developing unbiased and general rules:* As explained earlier different companies develop their own standards and practices that suit best their production plant and local conditions. Also different experts develop their own preferences and choices through years of experience. Hence, the devised processes and rules by experts reflect a combination of personal and company preferences and can be as narrow or broad as the experts' experiences.

These shortcomings were recognized in this work and different techniques were used to mitigate the impact on the quality of the research outcome. Process maps for each product type was developed that identify the type of information required for features and functions composing a product. In the product information process maps, modules of rule sets required for each feature and function were identified. To develop the rule sets, various design conditions were devised and presented to experts and the design process adopted by experts were traced and recorded. This process resulted in decision trees that represent the rules used in the decision-making process.

The developed process maps and decision-trees were reviewed by experts representing different segments of the industry to identify other potential decision paths. The knowledge acquisition process, of course, is a repetitive cycle where each cycle involves modification- expansion, deletion and change - of previously developed decision-trees and process models and development of new ones.

3.3 Reasoning Layer: Rule Library and Inference Engine

As shown in Figure 6, the reasoning layer is structured by developing domain-specific modularized rule libraries for various functions (e.g. connections, reinforcement, finishing, etc.) of different precast concrete product types (e.g. columns, beams, slabs, etc.). Rule

libraries are being developed using different inference mechanisms to infer new knowledge for QTO and CE of different aspects of a product. These rules have been applied on the information extracted from 3D parametric design models as well as user inputs regarding company limitations and preferences. We will use a combination of generic inference tools many times found as off-the-shelf inference shells and specific purpose reasoning modules developed for domain applications.

These modules represent the rules and reasoning for three major purposes semantic enhancement of design, task-based design evaluation and preparation and automated detailed design explained in detail in chapter three. These goals are achieved by developing rule sets that from the functional and operational aspects can be categorized in five major categories as follows:

- **Geometric and non-geometric attribute discovery and enrichment rules.** In the IFC schema each object type is defined by a minimal number of mandatory attributes and a larger set of optional attributes. Geometric attribute enrichment deals with:
 - (i) the optional attributes that although are defined in the data schema their values might be missing from an object definition in a design model. Object tag and *Description* are among these attributes.
 - (ii) attributes that their availability depends on the selected method of geometric representation: For instance in the solid extruded geometric modeling only the object profile geometry and extrusion length is available. In the boundary representation (Brep) method a solid object is created by a collection of

surfaces each of which defined by faces, edges and vertices. So surfaces are accessible in the Brep modeling but not in the extruded modeling.

(iii) attributes that are not part of the general object definition in the selected data schema but can be derived from the basic (geometric) information used to define an object. These range from simple object dimensions of width, length and height, to object surface areas and volume, to attributes related to specific features or variations of an object like blockouts, recessions and projections.

Value of these attributes when calculated and generated can be used only for the internal use in the chain of rules to reference features of an object and to filter and select objects based on features of interest and/or can be finally returned to the user by publishing them to the design model.

- **Attribute configuration rules.** These rules use a combination of logical and mathematical operations to configure new facts about geometry, topology, or other attributes of an object. These configurations are used to select objects from the work space to then apply other rules on them for property enrichment or predictive design.
- **Spatial topological relationship discovery.** These rules allow evaluating, discovering, expressing and referencing position, orientation and relation of one object or parts and features of an object relative to another object.
- **Object creation rules.** Design evaluation and advisory in our KBS often requires creation of new objects based on the existing objects with the advised attribute values. Some KBE systems provide the capability of geometry creation and manipulation. While geometry creation is out of the scope of this effort, this need

is fulfilled by creating logical objects and assigning to them the geometric and non-geometric attributes, necessary to perform the defined functions.

In the proposed system, object creation rules are typically developed to fulfill the automated detailed design goal. They usually use as input, results of a chain of rules in the above-mentioned categories which mostly aim at semantic design enhancement.

- **Object relationship creation rules.** Creating new object relationships enable building a hierarchical structure for the resulting enhanced and detailed design model. They are also important to reflect and communicate the design intent with the users which enables the users evaluate the inferred conclusions of rule sets and if necessary, to tweak the rules to achieve new results. Similar to new object creation rules, typically antecedents used in these rules are evaluated by previously discussed semantic design enhancement rules.

3.4 Validation

Various approaches have been deployed in this research to achieve higher levels of reliability about the acquired knowledge, developed rules and improving the current preconstruction practice. The major process includes:

- Development of product decomposition models illustrating information flow from various sources of knowledge for a selected class of building products. Various product models in different levels of detail were assessed.
- Development of a modularized library of rule sets that cover the complete set of functions from semantic enhancement of design to design evaluation and preparation for preconstruction tasks to automated detailed design

- Identification and formalization of a mapping between information flow requirements for automated design and sets of object model attributes and various classes of operators including geometric, spatial relationship operators.
- Implementation of the system by executing the developed rule sets on example test models representing identified design scenarios by domain experts. Verification of the results with the industry experts.

Approaches developed for each step to test and validate the results include:

- Assessing various product models developed in different levels of detail and developing all-encompassing models that represent various design scenarios and solutions.
- Identifying a wide range of design conditions by studying previous designs and introducing them to different experts and recording their thought process and rationale and variances in their selected approach.
- Identifying the differences in the QTO and CE process and rules and methods used by different industry experts.
- Generalizing the rules using parametric definitions in a way that different approaches and rules can be accommodated: the key is to define a minimum core industry-wide shared concept for each function and accommodate variances by parameters that users can select, tweak and adjust based on their local and company conditions and preferences.
- Validating the ultimate rules and their results with the experts representing different segments of the industry.

- Perform a comparative analysis between results achieved by execution of the rules in the developed KBS with similar design situations previously managed by traditional manual methods.

CHAPTER IV

SEMANTIC ENRICHMENT OF DESIGN MODELS: PRECAST CONCRETE COLUMN CASE

Adoption of a BIM-integrated workflow in the AEC projects will provide different project entities with the rich information embedded in parametric models to incorporate in various activities, improving their efficiency and potentially reducing errors and reworks and enhancing the accuracy of results. Streamlining such a workflow will enable exchanging and applying the information created by BIM platforms both horizontally across different entities working in parallel and vertically throughout the supply chain. Two main challenges for such a streamlined information flow throughout the AEC projects that haven't been sufficiently addressed by previous research efforts include lack of semantic interoperability and a large gap and misalignment of information between BIM information provided by design activities and the information required for performing preconstruction and construction activities. This research effort proposes a four stage framework for automatic semantic enrichment of design models and filling the information gap between design and preconstruction project activities. These four stages include development of product models, problem-solving algorithms, libraries of rule sets and a process for automatic addition of lacking information.

4.1 Introduction

Today's AEC projects involve many stakeholders and their collaboration on various aspects of design and construction is key for success of the projects [97]. Various studies [117, 118, and 119] have shown that collaborative practices lead to downsizing the errors and reworks and improving efficiency and productivity in creating, using and reusing knowledge throughout a project lifecycle. An efficient collaboration is only possible when sharing mechanisms for the created information by different entities from the contractual, cultural and technical standpoints are in place. While interoperability efforts and the resulted data model standard of the IFC [101] try to solve lexical and syntactic interoperability issues, semantic interoperability has remained to a large degree unsolved.

There is specially a lack of research in interpreting implicit semantics of design models, turning them to explicit facts and sharing them among different project stakeholders. The second unsolved problem is the large misalignment and gap between available information for design and analysis compared to required information for construction purposes. Based on the interviews conducted with several general contractors and subcontractors, this information misalignment and gap often imposes construction parties a complete rebuilding of models.

We conducted field studies interviewing with industry experts from several companies with a focus on using BIM for quantity take-off (QTO) and cost estimation (CE). This study showed that the current construction industry practice in these activities, especially in mid-sized and small construction companies, remains to a large extent, manual, error-prone and time-intensive, mostly relying on 2D drawings.

This chapter proposes a framework that attempts to use semantic enrichment of design models to provide semantic interoperability and fill the large information gap between normally available design model information at the end of the design development stage and information required for preconstruction and construction activities. The developed framework includes four major steps:

- (i) developing a product information model that identifies the information needed for preconstruction activities and information sources and processes that enable providing the required information;
- (ii) developing a problem solving algorithm to derive the required information from the available information;
- (iii) developing libraries of rule sets to implement the developed algorithm using a reasoning engine to infer new facts from the input data;
- (iv) developing a process to fill the remaining information gap between required and available model information in the enriched models

4.2 Multi-Party Collaboration and Semantic Interoperability

From the technical standpoint, collaborative design and construction means interoperability of software platforms that support various stages of design and construction activities.

Representation of information and information interoperability can take on various levels: encoding level, lexical level, syntactic level, semantic level and semiotic level [98]. Different industries have developed information sharing standards that to a large degree have resolved the interoperability issue in the encoding, lexical and syntactic level

[99]. EXPRESS language formalized in STEP (ISO 10303) [100] is one of the early interoperability standardization efforts. In the AEC/FM industry, the IFC data model [101], which was developed based on the EXPRESS language, with its comprehensive product modeling data schema and expandable structure is now the preferred interoperability solution. Although lexical and the syntactic interoperability are addressed by these standards, semantic interoperability still poses an enormous challenge. The model semantics comprise of two types of explicit semantics, directly expressed in design models and implicit hidden semantics [99]. Both explicit and implicit semantics are context dependent and refer to propositional meaning of the represented information [102].

IDM and MVD development efforts [103, 104] attempt to facilitate and standardize implementation of the IFC schema by establishing IFC bound concepts or units of information that are accepted industrywide to represent a standard interpretation of model semantics. The focus of IDM and MVD has been on defining explicit semantics and do not adequately address interoperability of implicit semantics.

4.2.1 Model Query Solutions

The first step for semantic interoperability of 3D parametric design models is providing the capability of querying spatial and non-spatial properties of objects. Querying models involves reading, extracting and analyzing the information relevant to the query subject.

Different Product Lifecycle Platforms (PLM) platforms aim in accommodating exchange of information among different project stakeholders. Central model management servers both proprietary and open source [105] are being developed to provide querying, integration of and leveraging shared information for different purposes.

A detailed review of major PLM systems, their querying and interoperability capabilities and a framework for integrating BIM servers with PLM solutions was previously provided by the author [106]. PLM solutions enable users to query a database for classified product (parts and assemblies) and project information mainly based on predefined criteria. Designed objects can be classified in various levels which is prerequisite for semantic queries. Semantic queries usually involve context-based and design-dependent classifications of objects which require semantic enrichment of design models. However, they do not perform semantic queries requiring object relationships interpretation.

Various methods have been developed to query model objects and index and retrieve them based on their geometric and topological similarity for use in the context of the manufacturing industry [107, 108, and 109]. However, in the building industry environment there have been a few efforts to analyze and query design objects and their relationships. One such effort [110] was built on the generalized model subset definition (GMSD) schema and aimed to filter models and build multi model views adding non-design related information to models. Another work [111] attempted to interpret implicit properties of objects and query model objects based on their predefined properties. Adachi developed a formal query language to use on the IFC-based models [112].

Analysis of spatial topological relationships of objects is an important aspect of querying BIM based 3D design models. A few research efforts have embarked on developing algorithms for analyzing topological relationships of 3D objects, and methods for implementation of spatial reasoning, for use in the AEC industry [113]. Most of these efforts either analyze each object individually and don't cover topological relationship analysis of objects or remain in theory and haven't been implemented.

4.2.2 Semantic Enrichment of Models

Semantic enrichment of models involves a rule-based expert system that utilizes a reasoning engine processing domain-specific rule sets and inferring new facts about model objects and their relationships and augments the models with these new facts [81]. Model information querying, filtering and retrieving based on object properties and relationships is an important prerequisite for the reasoning process and semantic enrichment of models.

In the context of mechanical assemblies and using PLM platforms, research efforts attempted to enrich models with information with functional and technological data of assembly features [114]; and, semantically enriched process models [115]. One of the few efforts for application in the AEC industry to semantically enrich models is called semantic enrichment engine for BIM (referred to as 'SEEBIM') [81]. SEEBIM reads and extracts variety of geometric and non-geometric object properties based on their IFC representation and is able to run spatial and non-spatial object relationships analysis on a simplified object geometry based on their bounding boxes.

4.3 Problem Definition

While Problem of interoperability, model query and semantic enrichment exist throughout projects' lifecycle, it is more complex and difficult to solve when there is a large shift in the model creation and application domains. In the AEC industry, this is specifically the cast when transitioning from design to preconstruction and construction stages. The type of information required in models and the useful way to represent this information is vastly different in design and construction stages. This creates a large gap between information creation and delivery in design development stages and information needs of construction

stages. While resolving interoperability issues can help the common sets of information created in design to be reused during the construction stages, it doesn't solve the misalignment and the gap between available and required information for preconstruction and construction purposes. Many times this information misalignment and gap imposes a complete rebuilding of models for construction parties that want to use BIM or limits the use and value of BIM.

Two of the major preconstruction activities that often additional information not available in design models and require different object representation methods are quantity take-off (QTO) and cost estimation (CE) activities. Our industry study show that due to these issues, the current industry practice in these activities, especially in small- and medium-sized companies remains to a large extent, manual, error-prone and time-intensive, mostly relying on 2D drawings.

One closely studied domain by the author is the precast reinforced concrete industry. In this industry, the units of quantity take off and cost estimation are precast concrete product pieces. However, the units of fabrication often are not distinguished in design models, which means elements like columns, slabs and wall panels are modeled as monolithic objects and not based on geometry of product pieces. This difference leads to rework and often for preconstruction purposes, different construction parties have to create their own models from scratch or as is the case in many of the interviewed companies abandon use of models and rely on 2D drawings.

4.4 Proposed Solution Framework

To resolve the problems discussed earlier and to streamline and semi-automate the use of BIM and BIM-based design throughout a project lifecycle, enhancing its value for project stakeholders a comprehensive framework is developed. This framework attempts to use semantic enrichment of design models to fill the large gap that normally exist between normally available design model information at the end of design development and information required for preconstruction and construction activities. The developed framework includes the following four major steps:

- *Develop Product models for precast concrete building elements.* This step relied on the knowledge acquired from domain experts and answers the ‘what’ question: (i) what design information is needed for preconstruction activities including QTO, CE and element fabrication; (ii) what design information is usually available by standard project contracts at the end of design development stage; (iii) what design information needs to be supplemented to enable automatic evaluation and preparation of BIM-based design model for preconstruction activities; (iv) what are the sources to acquire each information piece identified to be supplemented to design models; and finally (v) what rule libraries are required to guide the process of new information inference and addition to design models.
- *Develop a problem solving process for each cluster of information required to be supplemented to design models.* These groups of supplemented information will fill the gaps of available and required information for detailed design and construction of each product feature and for performing each step of the supply chain. This step like the previous one is based on acquired domain knowledge from experts and

answers the question of ‘How’ in a high level: These problem solving methods are in fact algorithms that demonstrate how through a set of successive steps that each use the outcome of the previous step they can supply the required information and fill the gaps between required and supplied design model information.

- *Develop rule libraries, each comprising of multiple rule sets.* These rule libraries together will solve the defined problems. This step answers the question of ‘How’ in a computer implementable level: In the rule libraries the pieces of data required to implement each step of the previously designed algorithms are identified; the data processing logic is defined; the data processing operators needed are defined and implemented; and finally the inferred information resulted from running the rule sets are added to design models to semantically enrich them.
- *Develop a path to provide the user with the lacking pieces of information required for automated detailed design but not included in the semantically enriched models.* This tracks both the initially available information in models and the supplemented design information and finds out how they can be used to derive the value of the yet unavailable pieces of information and complete the process of automated detailed design.

The next chapter sections delve deeper into each of these four major steps and will provide examples developed and implemented for each.

4.5 Product Model: Precast Concrete Column

Product models provide generic representations of different product types in the domain of discourse and build the core of knowledge bases. Various standards like the Unified

Modeling Language (UML) have been developed and used for structure, behavior and function modeling of products. Design and analysis rules can be embedded in product models to generate various design configurations of a specific product type. Figure 4.1

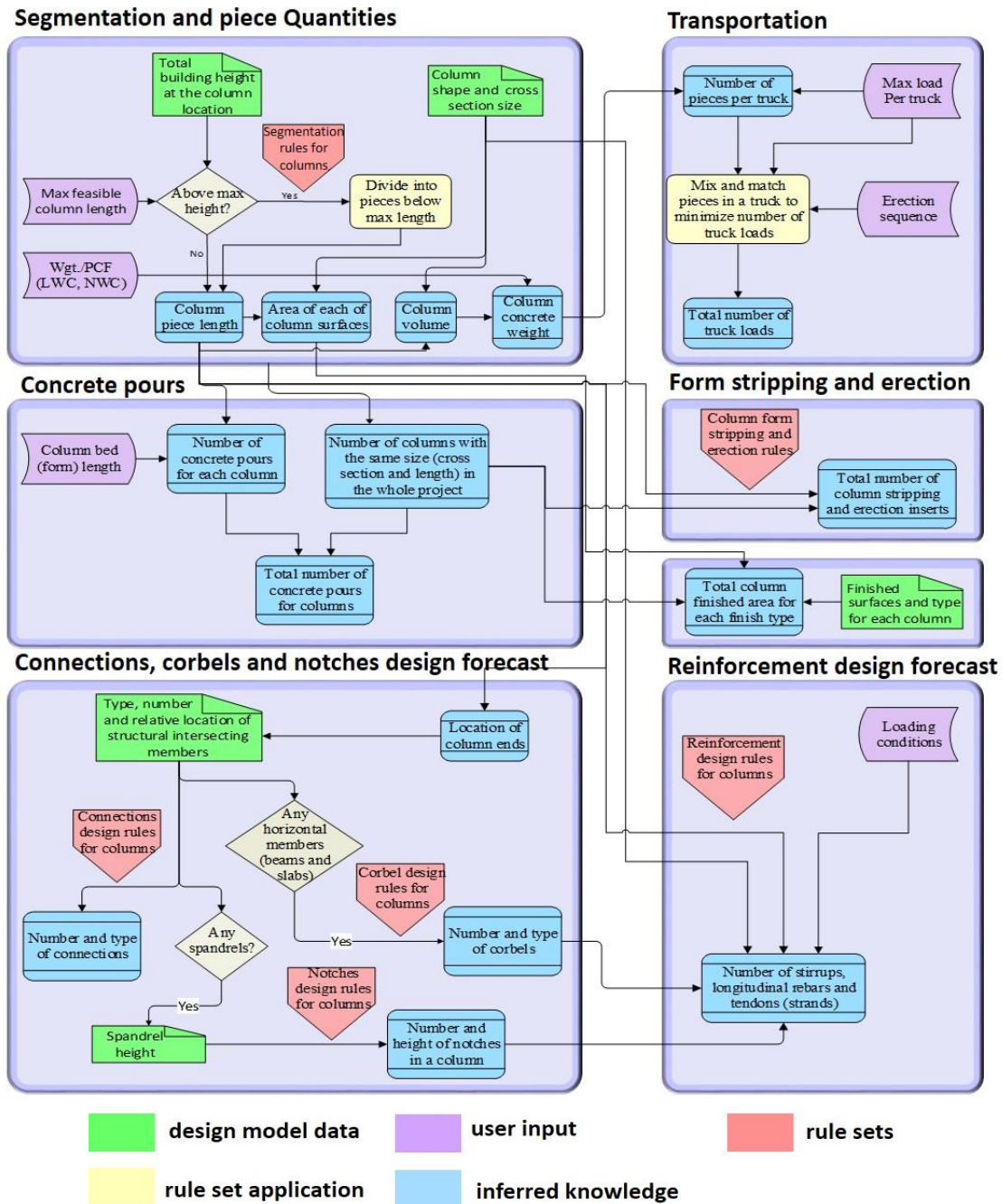


Figure 4.1: A precast concrete column product decomposition and information flow model

illustrates the product model developed for precast concrete columns.

The mainstream notations were studied and a customized notation was developed to demonstrate different types of information and information processing required for various stages of precast concrete column design and fabrication. The main goal of product modeling here was to

- (i) identify what types of inputs are required to infer new knowledge about different product features and to carry out diverse tasks throughout the supply chain of a product;
- (ii) identify information sources to acquire those essential inputs. These information sources include parametric design models, users and domain experts and are color coded in Figure 4.1; and
- (iii) identify essential rule sets to be developed to perform information processing and inferring new knowledge.

The nature of the information inputs cover a wide range from dynamic (e.g. design model data that is project-specific and often even changes throughout a project lifecycle) to relatively static (e.g. modularization rules based on architectural and structural design and supply chain rules) that can usually be considered fixed at the company level until new standards, products, or production technologies are adopted by the industry and the company at which point they need to be refined.

4.5.1 Problem Solving Methods and Knowledge Roles

The base of the developed rule sets is the notion of Problem Solving Methods (PSMs) and their knowledge roles. PSMs represent dynamic reasoning knowledge and make the

interactions between knowledge and problem solving processes and assumptions explicit [95]. An example of application of a PSM and different knowledge roles is illustrated in Figure 4.2. In this example if the value “60” is extracted from a design model as an observable for “total building height”, it can be abstracted to “above max height” using another observable of “50” which is a user input for “max feasible column height”. This abstracted observable will be followed by applying a solution abstraction of “divide to pieces below max length”, which will produce the solution of “column piece length”.

Note that to generate this solution, the PSM requires another set of inputs which are “segmentation rules for columns”. These rules are the end results of the process of knowledge acquisition and representation from domain experts. These rules themselves comprise of a cluster of PSMs that define the actions and the rationale behind each action and their implementation might require additional inputs.

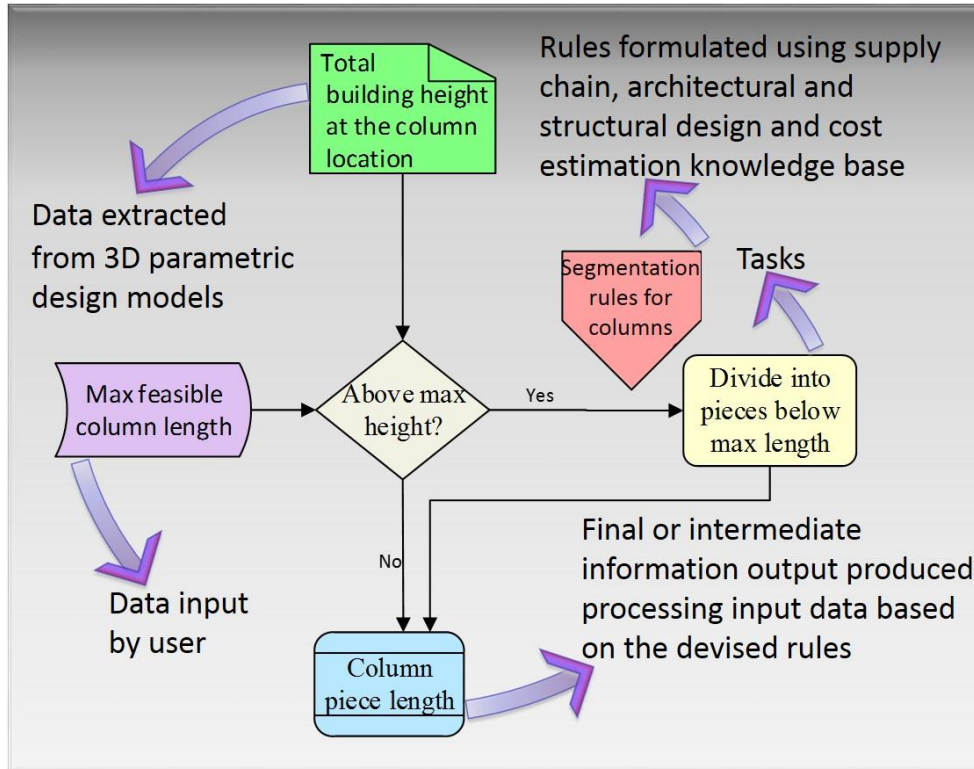


Figure 4.2: Example of a problem-solving method structure: inputs, outputs and actions

4.6 Development of a Problem Solving Algorithm

Based on the PCI design handbook [86], the goal of precast concrete modularization is to achieve minimum number of product pieces which will minimize the cost of production, transportation and erection. Therefore, the dimensions of each piece and its weight should be maximized within the constraints dictated by numerous supply chain related factors from plant layout to available form sizes, to truck and crane capacities, and transportation rules and crane access on the site. For precast concrete columns cross section profile is determined by loading conditions and structural analysis. So the main factor that maximizes size of columns is segmenting columns in a way that piece length is the closest possible to “max feasible column length” provided by the user.

Based on the interviews with industry experts, another criteria in segmenting columns, when the column length exceeds the max feasible column length, is to segment the columns in a way that lengths of the segmented column pieces are close to each other. That way the pieces can share most of the same features. Therefore, location of the column splices is determined relative to middle of the column.

Figure 4.3 shows the algorithm developed for segmenting precast concrete columns and inferring size and other attributes of suggested column pieces based on the acquired domain knowledge. Depending on whether the column is an internal or an external column, one of accessibility or aesthetics rules governs segmentation of the columns. If the column is an internal column the main criteria for the exact location of column splices are accessibility and comfort of working on connections between those columns. Hence, the column splices are usually located about 1.5'-2' above the finished floor.

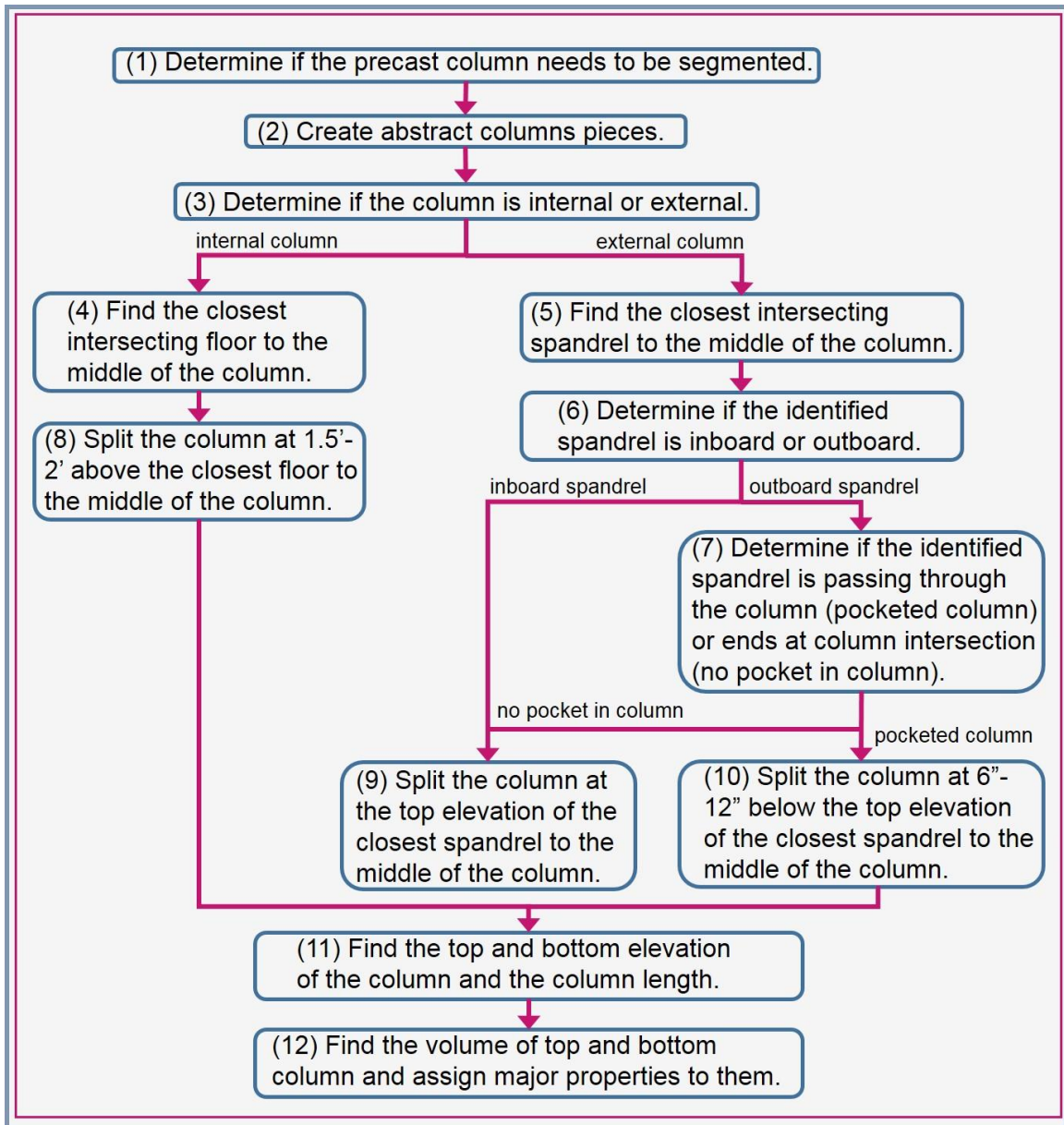


Figure 4.3: The algorithm developed for precast column segmentation

If the column is an external column the determining criteria are related to aesthetics. When the spandrel intersecting the column is outboard meaning that its exterior vertical wide surface is aligned with exterior vertical surface of the column and the column is pocketed (Figure 4.4 (a)), and it is possible to hide the connection behind the spandrel, the column

splice is located in 6"-12" below the top of the spandrel. When the intersecting spandrels are inboard, locating the exterior vertical wide surface of a spandrel aligned with the interior vertical surface of the column and the column is pocketed, hiding the column joints is not feasible. Then aesthetic criteria suggest locating the column joints in line with the top of spandrel. In the third situation the spandrel edge ends at the column edge which means the spandrel is not passing through the column and the column is not pocketed (Figure 4.4 (b)). In these cases, both when the spandrel is inboard and outboard, hiding the column joints is not feasible and the column splice location will be in line with the top of spandrel.

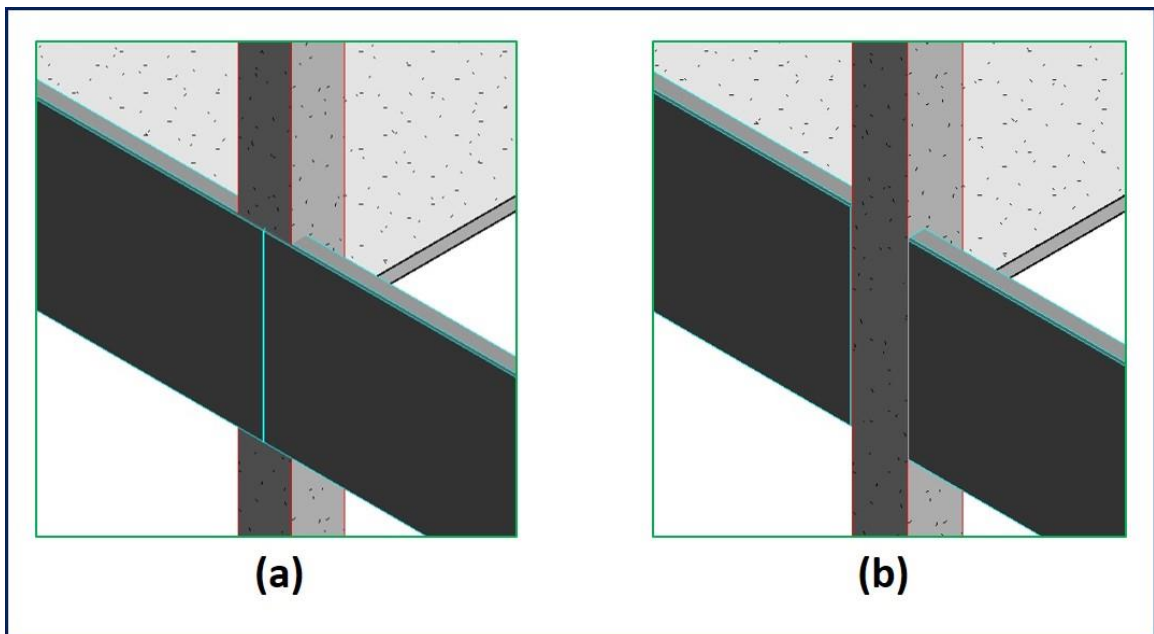


Figure 4.4: Relative spandrel-column positions; (a) outboard spandrel & pocketed column; (b) outboard spandrel but no pockets in the column

4.7 Rule Set Development and Semantic Model Enrichment for Column

Segmentation

The goal here has been to implement the algorithm demonstrated in Figure 4.3 by developing a set of rules to ultimately infer all the knowledge the user needs to segment the precast concrete columns. These rules use design information and user criteria and work in succession and each produce an output that is used as an input for the next rule.

The rules are defined as sets of logic rules that use the IF-THEN conditioning to infer a conclusion. Rule sets are related to each other through forward chaining. So the output of each rule is used as an input for the next rule. The inference engine searches and triggers all the rules whose antecedent match the available data queried from the design model data. The rule matching and activation progresses in cycles where in each cycle the rules in a rule set library are scanned sequentially to determine the ones match with the known facts. When passing through each rule, it scans all the objects in the model for finding the matches. Initiation of each rule results in inferring new facts determined in the ‘Then’ part of the rule. The facts inferred by running the matched rule in the first cycle will be added to the knowledge space. Once new facts are found by a rule, another cycle starts and the rule engine scans the rules once again since the new facts might provide the input required for initiation of other rules. These cycles proceed until no new fact is found.

4.7.1 SEEBIM Adoption

This research effort has adopted a rule inference engine called SEEBIM [81]. The rule engine is specifically developed for the AEC industry applications. In SEEBIM design model data is parsed using an open source product called IFC Viewer [83]. SEEBIM provides the capability of working with variety of geometric and non-geometric object

properties based on their IFC representation. It is also able to run variety of spatial and non-spatial object relationships analysis on a simplified object geometry based on the object bounding box. Examples of spatial topological operators include objects adjacency, contact, containment, and alignment. List of all the object attributes and operators used in this research effort are presented in Table 1. Many of these operators in this list had already been implemented by SEEBIM developers and tested on other use cases. The rest were identified, defined, developed and tested during this research work. In Table 1, the left column shows the attributes and operators available in SEEBIM before this research work and those added collaboratively during this effort.

4.7.2 Rule Structure

The developed rule sets use a set of geometric and non-geometric object attributes. These attributes and their values are either directly extracted from design model data or are calculated using the extracted model data. The developed rules also employ a set of spatial topological and non-spatial operators as well as mathematical and logical operators.

All the attributes and operators that are used in the column segmentation rule set is presented in Figure 4.5.

These attributes and operators provide the underlying structure of the rules: Different rules are built by mixing and matching these operators to apply task related analysis on selected objects and make certain conclusions. Rules are developed based on the following structure:

- Select maximum of two objects based on geometric and non-geometric properties (boxes with yellow heading in Figure 4.5) using the object attribute analysis operator of *is*.

Categories of Attributes and Operators	Previously Existed Attributes & Operators in SEEBIM	Attributes & Operators Added to SEEBIM in this Research
Geometric & Non-Geometric Properties Extraction and Calculation	length width height vertical_wide_faces vertical_narrow_faces Name Tag Description ObjectType ElementType Material	top_elevation bottom_elevation centroid_elevation aspect_ratio horizontal_top_face horizontal_bottom_face
Spatial Topological Operators	is_adjacent_to have_adjacent_faces is_in_contact is_overlapping_with do_objects_between_exist	is_closest_in_specific_direction have_aligned_faces do_not_have_aligned_faces
Mathematical and Logical Operators		evaluate_dimension compare_elements_attributes calculate_aspect_ratio measure_adjacent_dimension_of_elements measure_overlapping_dimension_of_elements compare_elements_attributes_in_lists
Non-Spatial Relationship Creation and Deletion Operators	create_aggregation_relationship create_association_relationship create_connection_relationship delete_relationship add_object_to_relationship	create_list_of_new_relationships add_object_to_list_of_relationships
Abstract Object Creation Operators	create_new_element	create_a_set_of_split_objects create_set_from_element create_list_of_new_elements
Non-spatial Property Value Designation Operators		set_element_attribute set_list_of_elements_attributes set_attributes_list_of_elements_in_relationship set_attribute_list_of_relationships
Non-Spatial Elements Relationship Analysis Operators	is_related_to is_not_related_to is_part_of / belongs_to is_not_part_of get_related_objects get_relating_object	find_relationships_containing_element
Non-spatial Property Analysis Operators	(element attribute) is (element attribute) is_not is_made_of filter_objects_between	

Table 4.1: comparative list of attributes and operators in SEEBIM used in this research work

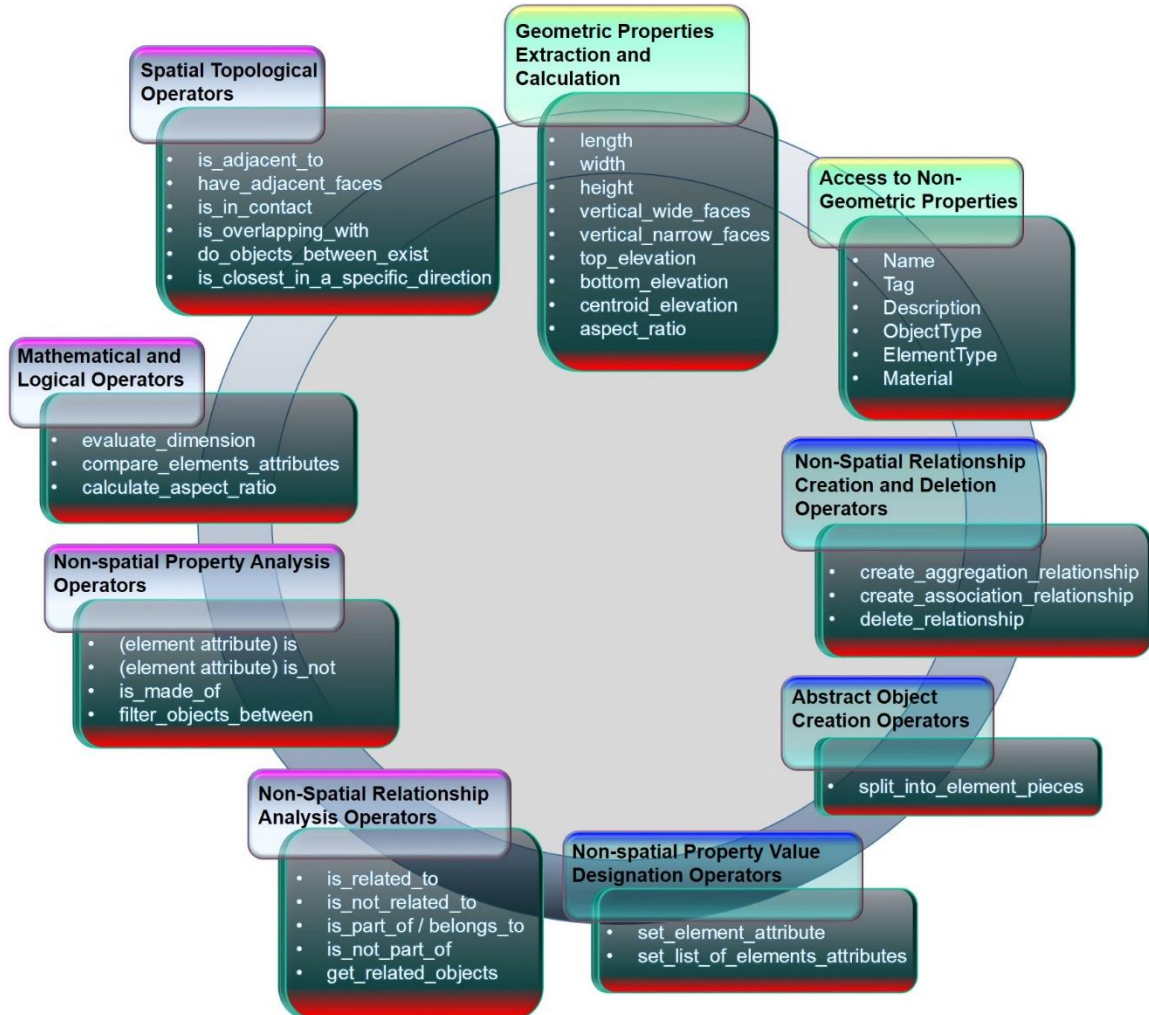


Figure 4.5: List of attributes and operators used in the column segmentation rule set

- Evaluate the selected objects based on predefined criteria applying a subset of the following operator categories (boxes with purple heading in Figure 4.5):
 - mathematical and logical operators
 - spatial topological operators
 - non-spatial property analysis operators
 - non-spatial elements relationship analysis operators

- Derive the conclusions and add the inferred knowledge to design models data using a subset of the following operator categories (boxes with blue heading in Figure 4.5):
 - non-spatial relationship creation and deletion operators
 - abstract object creation operators
 - non-spatial property value designation operators

These attributes, operators and their functions are explained in the next sections.

4.7.3 Rule Function Categorization

In a very high level and in terms of the end goal that various rules serve in the column segmentation use case, they belong to one or more of the following categories:

- design evaluation
- semantic design enrichment
- automated design for preconstruction and construction tasks

```

IF
<object1_ElementType> is 'lfcColumn' AND
<object1> is_made_of 'precast_concrete' AND
evaluate_dimension <object1_height> is '>' '50' AND
evaluate_dimension <object1_height> is '<=' '100' AND
<object1_domain_type> is_not 'column_segmented'
THEN
set_element_attribute 'object1_domain_type' set to
'column_segmented' AND
create_a_set_of_split_objects <object1> split into '2' pieces AND
set_elements_attributes created pieces 'Name', 'Description', 'domain_type',
'split_pieces'

```

Figure 4.6: column length evaluation and split pieces creation rule

object so that the user can examine which column pieces belong to which parent column.

A uniform color code for different components of rules is used in the pseudo codes representing the rules (Figure 4.7).

The rest of the rules in this rule set perform a combination of design evaluation and semantic enrichment to identify the best location on precast concrete columns for their segmentation. Semantic enrichment of a design is the result of some form of analysis on the extracted model data. Such analysis or reasoning attempts to discover the objects












	Spatial topological operators		Variables including objects and their attributes and tolerances
	Non-spatial relationship analysis operators		Values assigned to variables
	Non-spatial relationship creation and deletion operators		Connectors interjected for the better pseudo-code readability
	Non-spatial property analysis operators		Boolean constants
	Non-spatial property value designation operators		
	Abstract object creation operators		
	Mathematical and logical operators		

Figure 4.7: Color code legend for rules

properties and relationships implicit in design models and make them explicit. This fact discovery process is called knowledge inference and is followed by semantic design enrichment that communicates the new facts to users by adding them to the design model.

4.7.4 Geometric and Non-Geometric Attributes Extraction and Discovery

In the IFC schema each object type is defined by a minimal number of mandatory attributes and a larger set of optional attributes. The subset of attributes in this group that are extracted and used in the current rule set are listed under “non-geometric properties” in Figure 4.5 and are capitalized. These attributes are directly used as extracted from design models.

Geometric and non-geometric attribute analysis and discovery deals with the following scenarios:

- (i) the attributes that are defined in the IFC data schema, yet are optional and their values might be missing from an object definition or might not reflect the exact object properties or specific and sometimes domain-specific properties of significance for solving the defined problem. *ObjectType*, *Tag* and *Description* are among these types of attributes. For example whether a column is internal or external is of importance for segmenting columns. In our rule set after inferring the facts about this property for each column, the object *Tag* property is set to the inferred fact with respect to the column being internal or external (e.g. *internal_column*).
- (ii) attributes that their availability depends on the selected method of geometric representation: For instance in the solid extruded geometric modeling only the object profile geometry and extrusion length is available. In the boundary representation (Brep) method a solid object is created by a collection of surfaces each of which defined by faces, edges and vertices. So surfaces are accessible in the Brep modeling but not in the extruded modeling.
- (iii) attributes that are not explicitly definition in the IFC data schema but their value are derived from the basic geometric and topological information used to define the design model objects. These attributes range from simple object dimensions of width, length and height, to top, bottom and centroid elevation of an object and to object faces like *horizontal_top_face* and *horizontal_bottom_face*.

Value of these attributes when calculated and generated are used for the internal use in the chain of rules to reference, filter and select objects based on features of interest. When needed, these values are returned to the user by publishing them to the design model.

4.7.5 Functional Categorization of Operators

- *Spatial topological operators.* These operators allow evaluating, discovering, expressing and referencing topological relationships of two objects in the design model. The concept of bounding box or minimum bounding box is used to define and implement all the topological operators. It is important to note that in this work always when relationships like adjacency, overlap, containment and alignment between two objects are considered, they are defined and examined between bounding box of two objects. Hence, features like recession, blockout or dap in an object won't impact its relationships with other objects since they don't impact the geometry of its bounding box. The experience of solving several problems using the system has shown that simplifying object geometry to its bounding box sometimes have been helpful for solving problems and sometimes didn't provide complete information about an object needed to solve a problem and required developing a workaround.

An example of topological operators that is used in several of the developed rule sets is *is_adjacent_to*. Adjacency is defined here between two selected faces of two objects [81]. The definition below is provided in reference to Figure 4.8 where vertical wide face of object1 (F1) is adjacent to vertical narrow face (F2) of object2 within the given tolerance of D.

Two objects have adjacent faces in a given tolerance limit if

- when two specified faces ($F1$ & $F2$) of the objects' bounding boxes are projected on two perpendicular axes that are parallel with those object faces ($axis1$ & $axis2$), have a common projected surface area of larger than zero, and;
- other surfaces of the two objects when projected ($pf1$ & $pf2$) on parallel axes ($axis3$) are disjoint and have no common surface area, and;
- the distance between those projected surfaces on parallel axes to those faces ($pf1$ & $pf2$) is within the selected tolerance limit ($D \leq \text{tolerance}$).

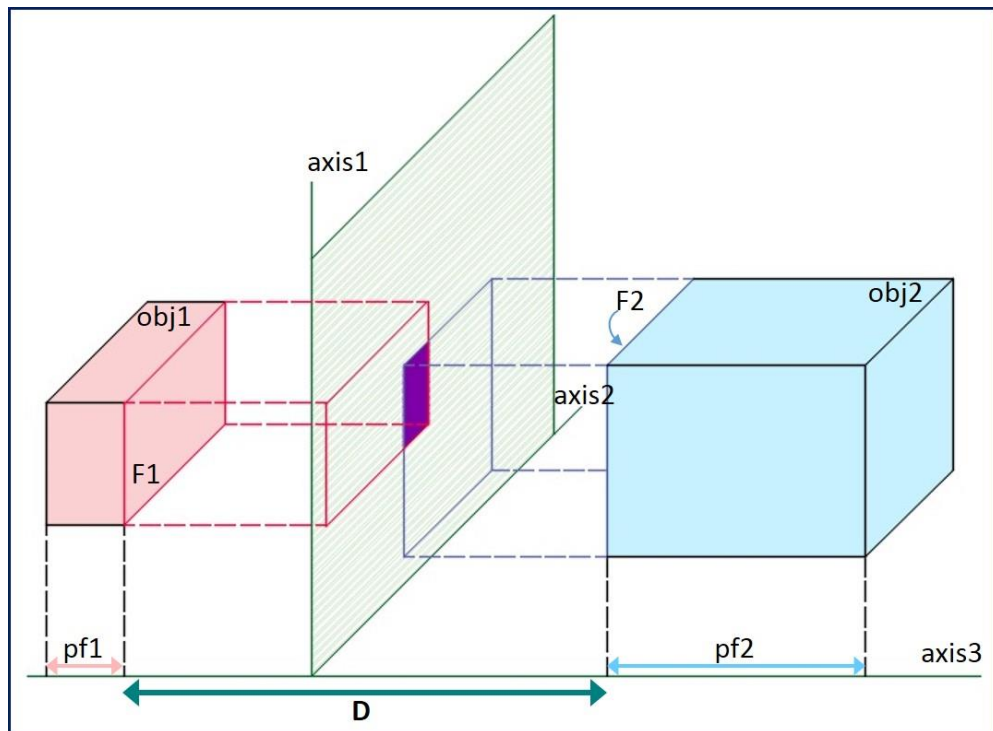


Figure 4.8: Implementation of two objects' adjacency relationship analysis operator

are the proximate surface area created by extending and projecting one of the first object's faces within a given tolerance limit and in the surface normal direction over second object is larger than zero.

- *Mathematical and logical operators.* They are employed to evaluate dimensions of an object using logical operators or compare attributes of two objects to each other. Examples of their application is provided later in object classification rule examples.
- *Non-spatial property analysis and value designation operators.* The operators *is*, *is_not* and *is_made_of* examine the non-geometric object attributes to find out if they are equal or unequal to a certain value. The operator *filter_objects_between* selects an object located in the proximate volume between two objects based on a specified object attribute value. This enables referencing an object between two selected objects in a rule and performing an operation on it.

Using property value designation operators in the consequent clauses of the rules, a certain value is assigned to an attribute or set of attributes of an object that meets the specified conditions in the antecedent part of the rule.

- *Non-spatial relationship analysis, creation and deletion operators.* In the IFC data schema various relationships can be defined between objects. These relationships are subtypes of the abstract entity of *IfcRelationship*. Relationships serve different purposes like creating a hierarchical building structure in a design model among different objects, or connecting objects to each other. The relationship creation and deletion operators can create or delete any number of relationships between objects. The elements relationship analysis operators can examine existing relationships in

design models or relationships created by the rule engine and can be further divided into 3 groups:

1. *is_related_to* and *is_not_related_to* examine the existence of a selected relationship among two objects
2. *is_part_of / belongs_to* and *is_not_part_of* examine whether an object has participated in a relationship as a related or relating object or a realizing element.
3. *get_related_objects* and similar operators provide access to objects in a specified relationship with the selected objects in a rule.

As mentioned earlier in each rule a maximum of two objects can be selected. Yet many complex design situations involve several objects and require getting access and operating on other objects that are related to the main selected objects. The second and third group in relationship analysis operators' category as well as *do_objects_between_exist* in spatial topological operators' group and *filter_objects_between* in non-spatial attribute analysis operators' group are valuable in these situations. They enable selecting and explicitly referencing objects related to the two selected objects in the rules. In section 4.8.1.2 one example of such scenarios will be discussed.

- *Abstract object creation operators.* Design evaluation and advisory often requires creation of new objects based on the existing objects with the advised attribute values. While geometry creation for new objects is out of the scope of this effort, this need is fulfilled by creating logical objects and assigning to them the geometric and non-geometric attributes necessary to be counted as the intended type of objects

absent from the design. The function of *create_a_set_of_split_objects* operator that belongs to this category was earlier explained. In the proposed system, object creation operators are typically developed to fulfill the automated detailed design goal.

4.8. Example Rule Sets: Structure Analysis and Results

4.8.1 Object Classification Rules

These rules attempt to classify the designed objects based on specific aspects of their design which are of interest to solving the problem at hand. Three sets of rules in the rule set library created for column segmentation deal with object classification. The results of these rules are used to narrow down the work space and select specific instantiations of an object class for further analysis, semantic enrichment or automated design purposes.

4.8.1.1 Beam Classification

Both spandrels and rectangular, L-shape and inverted-tee beams are structurally considered as beams and are extracted from BIM authoring tools as *IfcBeam*. Yet for many different purposes, we need to distinguish spandrels from other types of precast concrete beams. In the column segmentation problem, this classification provides an input for rules in step 3-10 of the algorithm presented in Figure 4.3. The best geometric metric to distinguish spandrels from other beam types is the aspect ratio. Aspect ratio for structural members is generally defined as the profile height divided by width. Figure 4.9 demonstrates the range of width and height dimensions of major precast concrete beam types and spandrels [86]. Since in some very deep non-rectangular beams, ratio of the top surface width to beam's

height can get very close to spandrels' and to avoid any overlap the rule is defined to use the bottom surface width in the formula.

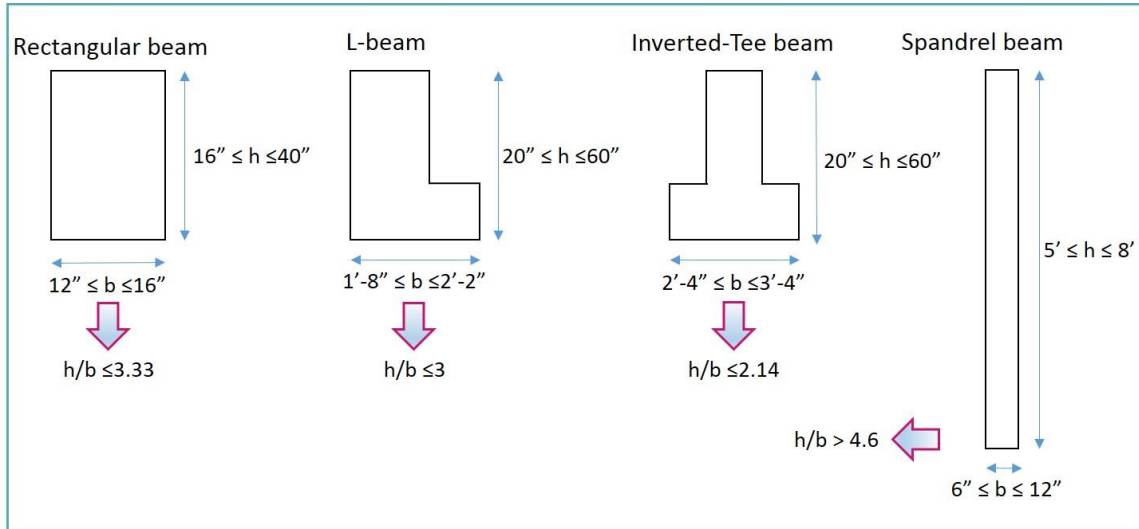


Figure 4.9: Range of width, height and aspect ratio for different types of precast concrete beams

Figure 4.10 represents spandrel versus non-spandrel beam classification rules. After defining the concept of aspect ratio and calculating it for the selected object, the result is examined and the appropriate classification is assigned to the object *Tag*. In figure 4.18 these values can be seen in the resulted enriched IFC file opened in the Solibri Model Viewer [116]. The same results can be viewed when the enriched file is opened in Autodesk Navisworks Manage software under object properties. They are shown under the BATID attribute which is the Solibri equivalent for *Tag* attribute in IFC.

4.8.1.2 Internal and External Column Classification

Other examples of classification rules developed for the algorithm depicted in Figure 4.3 are rules to define and distinguish between internal columns, external columns and a third type that we refer to as *segmented_like_internal_column*. The reason for this type of classification and how it helps to solve the column segmentation problem is discussed in

```

IF
<object1_ElementType> is 'lfcBeam' AND
<object1> is_made_of 'precast_concrete' AND
calculate_aspect_ratio <object1>, 'height / weight' AND
evaluate_dimension <object1_aspect_ratio> is '<' '4.6' AND
<object1_Tag> is_not 'non_spandrel_beam'
THEN
set_element_attribute <object1_Tag> set to 'non_spandrel_beam'

IF
<object1_ElementType> is 'lfcBeam' AND
<object1> is_made_of 'precast_concrete' AND
calculate_aspect_ratio <object1>, 'height / weight' AND
evaluate_dimension <object1_aspect_ratio> is '>=' '4.6' AND
<object1_Tag> is_not 'non_spandrel_beam'
THEN
set_element_attribute <object1_Tag> set to 'spandrel'

```

Figure 4.10: Rules to identify spandrels and non-spandrel beams

section 4.6. The following definitions are developed for this classification, based on the acquired knowledge of domain experts:

```

// If a precast concrete column is intersecting with at least 2 beams at the same
// floor level the column is an internal column.//

//If a precast concrete column is intersecting with a spandrel or a wall panel
// (building cladding element) the column is an external column.//

//If a precast concrete column is intersecting with a spandrel as well as at least 2
// beams at the same floor level the column needs to be segmented like an internal
// column.//

```

The rule developed for identifying internal columns is depicted in Figure 4.11. First beams were classified and two non-spandrel beams were selected. To find out if they were in the same floor their top elevation was examined. Then the selection pool was narrowed down to those beams that were adjacent within the distance that equals a column's width. Most often width of the precast concrete columns is below 3', hence this distance was given as the tolerance number for adjacent faces, but depending on the largest column size used in a given project, and if needed, this number can be changed by the user. At this point, the proximate volume between the two selected adjacent beams was evaluated and the selection pool was further narrowed down to those beam pairs that had an object in between with an *ElementType* equal to *IfcColumn*.

```

IF
<object1_ObjectType> is 'non_spandrel_beam' AND
<object2_ObjectType> is 'non_spandrel_beam' AND
compare_elements_attributes <object1_top_elevation> '='
    <object2_top_elevation>, 'tolerance' '0.1' AND
have_adjacent_faces <object1_vertical_narrow_faces> is adjacent to
    <object2_vertical_narrow_faces>, 'tolerance', '3' AND
do_objects_between_exist <object1>, <object2> AND
filter_objects_between_based_on 'ElementType' equal to 'IfcColumn' AND
filtered_objects_between 'Tag' is_not 'segmented_like_internal_column'
<object1> is_not_related_to <object2> relationship of 'domain_type',
    'adjacent_beams_same_floor' AND

THEN
create_classification_relationship 'IfcRelAssociatesClassification', between
    <object1>, <object2>, 'Name', 'adjacent_beams', 'Description',
    'adjacent_beams_with_column_in_between',
    'domain_type', 'adjacent_beams_same_floor' AND
set_list_of_elements_attributes_filtered_objects_between 'Tag' set to
    'internal_column'

```

Figure 4.11: Internal column classification rule

In the next step *Tag* of the column between the beams was examined. As the definition of columns to be segmented like an internal column shows, these columns meet all the conditions of internal columns plus they are intersecting spandrels. Thus, and as demonstrated in Figure 4.12, after columns were identified as internal columns, they were examined in the next rule to see if they were adjacent to or overlapping with a spandrel. If they were, their *Tag* values were changed accordingly.

```
IF
<object1_Tag> is 'internal_column' AND
<object2_ObjectType> is 'spandrel' AND
(<object1> is_adjacent_to <object2>, 'tolerance', '0.1' OR
<object1> is_overlapping_with <object2>) AND
<object1_Tag> is_not 'segmented_like_internal_column'
THEN
set_element_attribute <object1_Tag> set to 'segmented_like_internal_column'
```

Figure 4.12: Classification of columns to be segmented like internal columns

After a column is classified as *'segmented_like_internal_column'*, and in the next cycle the same column can be selected in the internal column rule and since it meets internal column conditions its *Tag* will be changed to *internal_column*. This cycle will then be repeated and it can create an infinite loop. To avoid this problem, *Tag* of the column was examined to make sure it was not *'segmented_like_internal_column'*.

The last step in the IF clause of the internal column rule was to make sure that this rule had not already been applied to the selected beams. In the THEN clause a classification relationship with the *domain_type* of *'adjacent_beams_same_floor'* was created. Therefore, if this beam pair had earlier undergone this rule they should be related to each other by a relationship of this *domain_type*. Without this condition, this rule will be

executed on the same beam pair over and over again which will again create an infinite loop.

The first type of infinite execution loop that is caused when new facts found by one rule reactivates another rule is called a complex-loop. The second type of infinite loop that is caused by infinite times of execution of one rule is called a self-loop [96]. The strategies explained above to prevent these loops are used in most of the other rules and will be briefly referred from now on as no-loop control declarations.

In the THEN clause in addition to creating a relationship between the two beams, *Tag* of the filtered column is changed to *internal_column* to communicate this fact with the user.

Earlier in segment 4.7.4, it was mentioned that in some situations more than two objects need to be examined in a rule. The situation explained for internal column classification involved three objects: two beams and a column between them. As seen above, the two operators of *do_objects_between_exist* and *filter_objects_between* enabled access to the third object which is a column.

The last rule in this rule set is depicted in Figure 4.13 which classifies external columns. External columns intersect a spandrel or a wall panel and in each floor level intersect maximum of one non-spandrel beam. The intersecting columns and spandrels are either adjacent or overlapping each other (Figure 4.4) and these conditions are examined in this rule. In general columns intersecting a spandrel or a wall panel can intersect up to three beams in each floor. Hence the columns to be segmented like internal columns can also be selected by this rule as external columns which again creates an infinite loop. The last two lines in the IF clause provide no-loop control declarations to avoid the two types

of loops explained earlier. The THEN clause changes the column's *Tag* to reflect its classification.

```
IF
  <object1_ElementType> is 'IfcColumn' AND
  (<object2_ObjectType> is 'spandrel' OR
  <object2_ElementType> is 'IfcWallStandardCase') AND
  <object1> is_made_of 'precast_concrete' AND
  (<object1> is_adjacent_to <object2>, 'tolerance', '0.1' OR
  <object1> is_overlapping_with <object2>) AND
  <object1_Tag> is_not 'segmented_like_internal_column' AND
  <object1_Tag> is_not 'external_column'
THEN
  set_element_attribute <object1_Tag> set to 'external_column'
```

Figure 4.13: External column classification

Figure 4.14 depicts the results of the enriched IFC file where all three types of columns in the test model were correctly classified. Note that while in the context of this chapter, this classification is used as a guide for column segmentation, it is an important concept with broad applications. This classification is essential for design, production planning and construction of insulations, finishes, type of connections used and other aspects of building elements.

4.8.1.3 Pocketed and Non-Pocketed Column Classification

Step 7 of the algorithm depicted in Figure 4.3 deals with the type of intersection between a spandrel and a column and whether the column bounding box is overlapping the spandrel bounding box in which case the column is pocketed. This classification is achieved through

a simple rule where in its THEN clause the appropriate classification value is assigned to columns' *Description* attribute as depicted in Figure 4.14.

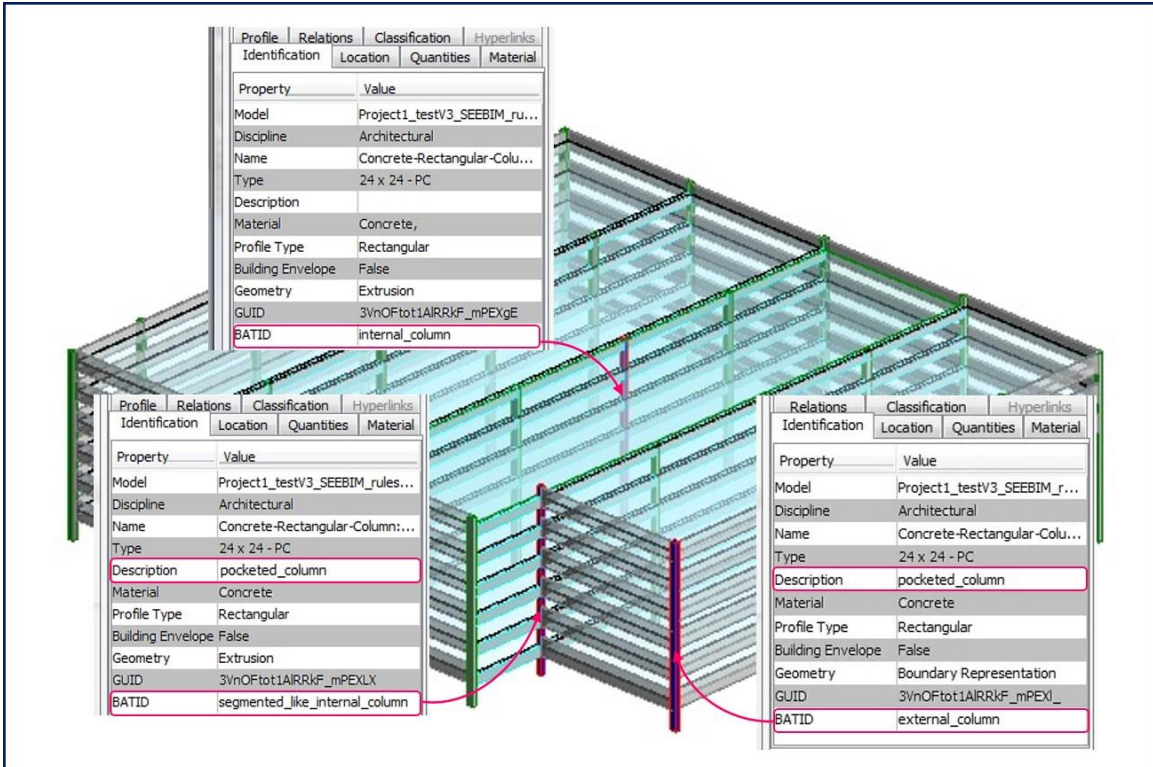


Figure 4.14: The enriched IFC model for column classifications

4.8.2 Rules to Find Closest Objects to another Object in a Specified Direction

This concept also has a broad applications and for example can be used to optimize locations of emergency exits in buildings. In the context of precast concrete column segmentation it is used to find the closest floor to internal and segmented like internal columns' centroid and the closest spandrel to external columns' centroid. To find the closest floor or spandrel to a column's centroid the distance of top elevation of floors and spandrels to column's centroid elevation in the vertical direction is compared. After figuring this out, columns will be segmented in an elevation relative to the closest finished

floor level or top of spandrel according to the algorithm in Figure 4.3. This will then enable users to calculate length of each column piece after splicing columns.

The rule set for this purpose includes four rules, two for finding the closest floor to internal and segmented like internal columns' centroid and two for finding the closest spandrel to external columns' centroid. The logic and rules' structure used for internal and external columns are similar; hence, only one set is depicted (Figure 4.15 and 4.16) and discussed in this chapter.

IF	
<object1_ObjectType> <i>is</i> 'non_spandrel_beam' AND (<object2_Tag> <i>is</i> 'internal_column' OR <object2_Tag> <i>is</i> 'segmented_like_internal_column')	(1) Select main objects based on their classifications
<object1> <i>is_adjacent_to</i> <object2>, 'tolerance', '0.1' AND	(2) Verify that the selected beam (B1) and column (C1) are adjacent
<object2> <i>is_not_part_of</i> relationship of 'domain_type', 'closest_intersecting_beam_column'	(3) Verify that the rule is only executed on each column once
THEN	
<i>create_association_relationship</i> 'IfcRelAssociatesClassification', <i>between</i> <object2>, <object1>, 'Name', 'Description', 'domain_type', 'closest_intersecting_beam_column' AND	(4) Create a relationship between the beam and the column
<i>set_element_attribute</i> <object1_Description> <i>set to</i> 'closest_intersecting_beam_to_column_centroid' AND	(5) Change the beam's tag to reflect the initial assumption that it is the closest to the column centroid
<i>set_element_attribute</i> <object2_ObjectType> <i>set to</i> 'column_checked_for_closest_beam'	(6) Change the column ObjectType to use as a control point

Figure 4.15: Rule I - Closest floor to internal and segmented like internal columns' centroid

The first rule (Rule I) in this group of rules and its steps is illustrated in Figure 4.15. In a nutshell, it creates an association relationship between each internal and segmented like internal column in the model (C1) and only one of the non-spandrel beams intersecting each column (B1). The operator *is_not_part_of* verifies that the selected column does not belong to a relationship with the *domain_type* of 'closest_intersecting_beam_column'. Since this relationship is created between the column and its intersecting beam after the

rule is executed once on any selected column, this operator doesn't allow more than one such a relationship to be created for each column. This rule makes an initial assumption that the randomly selected intersecting beam in Rule I is closest to the column centroid and changes the beam's *Description* attribute to reflect this assumption. This will be the case until the next rule in the group proves otherwise. Changing the beam's attribute will help if the randomly selected beam in this rule is in fact the closest beam to the column's centroid. In that case Rule II will not be triggered and the *Description* of B1 will remain as '*closest_intersecting_beam_to_column_centroid*'.

Note that to find the closest floor to a column's centroid both beams and slabs intersecting the column can be used in the rules since the top finished elevation of both in each floor is usually the same and the choice of the object1 in rules doesn't impact the rules' logic, only that slabs intersecting the column might either be adjacent to or overlapping with columns. Hence, in the case of using slabs in the rules one line should be added to examine whether the overlapping relation of two objects in addition to their adjacency relation.

The structure and steps of the second rule (Rule II) in this group is demonstrated in Figure 4.16. This rule is executed on all the internal or segmented like internal columns in the model that have already passed through Rule I and all their intersecting beams except the one passed through Rule I. The goal of the this rule is to find out whether there is another intersecting beam with the column that is closer to the column centroid and replace the beam found in the first rule with this closer beam.

In this rule again we need to have access and reference three objects: If we assume the rule is being executed for the nth time on a column, in addition to that column (C1), we

IF

<object1_ObjectType > <i>is</i> 'non_spandrel_beam' AND (<object2_Tag> <i>is</i> 'internal_column' OR <object2_Tag> <i>is</i> 'segmented_like_internal_column')	(1) Select main objects based on their classifications
<object1> <i>is_adjacent_to</i> <object2>, 'tolerance', '0.1' AND	(2) Verify that the column (C1) and the beam (B2) are adjacent
<object2> <i>is_not_related_to</i> <object1> <i>relationship</i> of 'domain_type', 'closest_intersecting_beam_column' AND	(3) Verify that the selected beam here (B2) is not the same one selected in Rule I (B1)
<object2> <i>belongs_to</i> <i>relationship</i> of 'domain_type', 'closest_intersecting_beam_column' AND	(4) Verify that the selected column has already been passed through rule I (C1)
<object1> <i>is_closest_in_specific_direction</i> in vertical direction to <object2_centroid_elevation>	(5) Compare the distance of the top elevation of currently selected beam to the column centroid (B2-C1) with the distance of the beam in the closest_intersecting_beam_column relationship and check if the current one is closer to the column centroid

THEN

<i>get_related_objects</i> in relationship of 'domain_type', 'closest_intersecting_beam_column' AND	(6) Find the beam related to column in rule I (B1)
<i>set_element_attribute</i> related object in relationship of 'domain_type', 'closest_intersecting_beam_column' <i>set</i> 'Description' to 'not_closest_beam' AND	(7) Change the domain type of this beam (B1)
<i>delete_relationship</i> 'closest_intersecting_beam_column' <i>found by</i> <i>the belongs_to operator</i> AND	(8) Remove the relationship created in rule I between the beam and the column (B1-C1)
<i>create_association_relationship</i> 'IfcRelAssociatesClassification', <i>between</i> <object2>, <object1>, 'Name', 'Description', 'domain_type', 'closest_intersecting_beam_column' AND	(9) Create a relationship between the closest beam and the column (B2-C1)
<i>set_element_attribute</i> <object1_Description> <i>set to</i> 'closest_intersecting_beam_to_column_centroid'	(10) Change the description of B2 to communicate to the user that this beam is closest to the column centroid

Figure 4.16: Rule II - Closest floor to internal and segmented like internal columns' centroid

need to have access to an intersecting beam selected in this execution (B2), and to either the beam undergone the previous execution of this rule (n-1th time) or if n=1, the beam undergone Rule I (B1). Whenever Rule I and II are executed on a column, a relationship of *domain_type closest_intersecting_beam_column* is created between the column and the beam involved in that rule execution. As mentioned earlier this rule is only executed on

columns that have passed through Rule I and are already part of such a relationship. The operator *belongs_to* used in step (4) of Rule II examines and verifies this condition. Moreover it enables us to get access to objects related to the column in this relationship. Hence in step (5) *is_closest* operator has access to both B1 and B2 and compares the distance of the top elevation of B1 and B2 to the column and finds out if B2 is closer than B1 to the C1 centroid. Only if the B2 is closer to the C1 centroid than B1, the rule will be executed.

In the THEN clause first the operator *get_related_objects* finds the objects related to the column in relationships earlier found by *belongs_to* operator, designated as B1. Since the execution of Rule II means that another beam found to be closer to the column than B1, in step (7) the operator *set_element_attribute* changes *Description* of the B1 to *not_closest_beam* and in step (9) builds a new relationship between C1 and B2. Finally in step (10) *Description* of B2 is set to *closest_intersecting_beam_to_column_centroid*.

When the execution of rules end, the beams with such a *Description* value will actually be the closest to their intersecting column's centroid; since it means that Rule II couldn't find any other beam closer to each column evaluated by Rule I and II, otherwise it would have changed its *Description* to *not_closest_beam*. Figure 4.17 illustrates the results achieved by running the rule sets on a test model for an example internal column and an example external column.

4.9 The Enriched IFC File Results and Final Phase

Figure 4.18 depicts a collection of snapshots taken from an enriched IFC P21 file that was the result of execution of all the rule sets explained earlier in this chapter. The test model

is a 6-floor building with 4 column bays in one side and 5 in the other side. Heights of the columns in the model are above 67' which means they exceed the maximum feasible column length set at 50' and are required to be segmented.

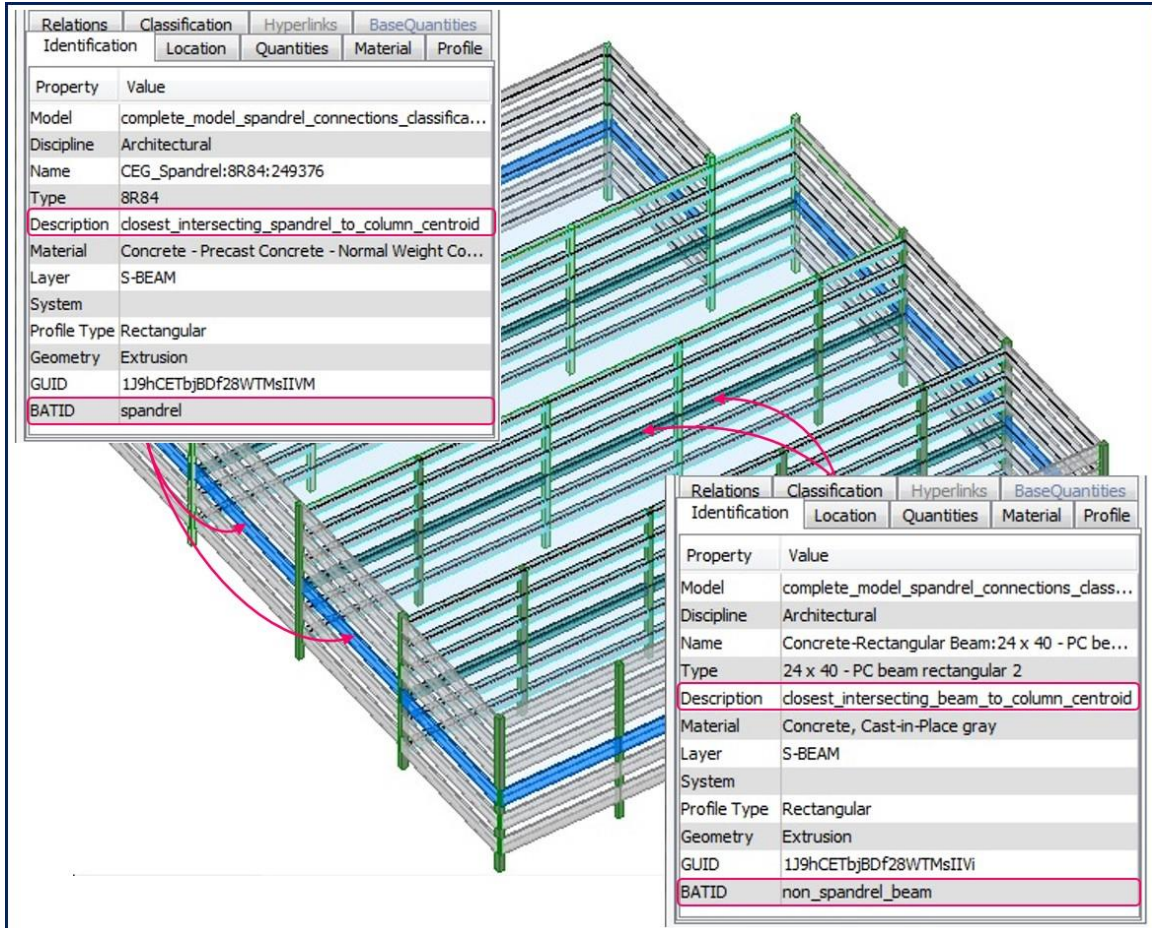


Figure 4.17: The enriched IFC model depicting closest floor/spandrel to columns as well as beam classification example results

One external column (#324), one internal column (#2046) and one segmented like internal column (#1357) is selected for these snapshots and their classification as internal or external is expressed in their *Tag* attribute values. Also for the pocketed columns this fact is reflected in their *Description* value. The value of '*column_segmented*' verifies that the columns have passed through the rule in Figure 4.6 that evaluates their length and it


```

9973 #324= IFCCOLUMN('0kYKsSUv16RPb4s_6_kT7R',#41,
9974 'Concrete-Rectangular-Column:24 x 24', 'pocketed_column', 'column_segmented'
-,#322,#315,'external_column');

9128 #17287= IFCBEAM('0mZls02EH14fpVE3F56dI6',#41,'SP:8R84',
9129 'closest_intersecting_spandrel_to_column_centroid', 'spandrel',
9130 '#17250,#17285,'159139');

7301 #2046= IFCCOLUMN('0kYKsSUv16RPb4s_6_kTvh',#41,
7302 'Concrete-Rectangular-Column:24 x 24', '$', 'column_segmented',
7303 '#2045,#2040,'internal_column');

6298 #11924= IFCBEAM('1x3h30XMv1nASITH1DaK7E',#41,'Concrete-Rectangular
6299 Beam:24 x 40', 'closest_intersecting_beam_to_column_centroid',
'non_spandrel_beam',#11902,#11922,'152833');

798 #1357= IFCCOLUMN('0kYKsSUv16RPb4s_6_kTvh',#41,
799 'Concrete-Rectangular-Column:24 x 24', 'pocketed_column', 'column_segmented'
-,#1356,#1351,'segmented_like_internal_column');

10262 #19340= IFCBEAM('1J9hCETbjBDf28WTMsIIUK',#41,'Concrete-Rectangular
Beam:24 x 40', 'closest_intersecting_beam_to_column_centroid',
'non_spandrel_beam',#19318,#19338,'165129');

15408 #30323= IFCRELAGGREGATES('0eZcZgRjf2EwPWn69jmwOn',#41,
15409 'closest_intersecting_beam_column', 'closest_intersecting_beam_column',
'#2046, (#11924));
15410 #30370= IFCRELAGGREGATES('3Binwjnd5DOvL1wMWclhlu',#41,
15411 'closest_intersecting_spandrel_column', 'closest_intersecting_spandrel_column',
'#324, (#17287));

15652 #29995= IFCRELAGGREGATES('2NCwJDh_nEZffBfaNr8Agm',#41,
15653 'closest_intersecting_beam_column', 'closest_intersecting_beam_column',
15654 '#1357, (#19340));

```

Name	Description	ObjectType	Tag
------	-------------	------------	-----

Figure 4.18: Collection of snapshots from an enriched IFC P21 file created by execution of column segmentation rule sets

has determined that the columns depicted here needed to be segmented and new column pieces are added to the IFC file. One of the three beams selected for the snapshots is a spandrel (#17287) and the other two are of *non_spandrel_beam* type (#11924 & #19340) and this fact is conveyed through their *ObjectType* value. Each of these selected beams is also closest to one of the selected columns, illustrated in their *Description*. As such in each of the *IfcRelAggregates* relationships created one of the column beam pairs that are closest

to each other participate; the column as the *RelatingObject* and the beam as the *RelatedObjects*.

The only step in the column segmentation algorithm not discussed in this chapter is step (6) that involves evaluation of spandrels to figure out if they are inboard or outboard. With the currently available operators, this proved to be a complex problem that involves many different design situations and requires several rule sets to solve. Due to this and also that being inboard or outboard is an important factor in determining the type of connections between spandrels and columns, the related rule sets to step (6) will be discussed in the chapter that discusses automated design of connections between precast concrete elements. Except that, all the information essential for determining the location of precast concrete columns' segmentation are provided by the rule sets discussed in this chapter.

The final steps to estimate the length of newly created column pieces based on the information added to the enriched model are as follows:

- If we assume that a column is segmented to two pieces, the bottom elevation of one of them (which will be referred here as bottom column) will be equal to the bottom elevation of the parent column and the top elevation of the other (which will be referred here as top column) will be equal to the top elevation of the column. Through the relationships created between the top and bottom column pieces and their parent columns, the top and bottom elevation of parent columns for each new column pair can be assigned to new columns' attributes.
- The next step is to find the top elevation of the bottom column piece and to find the bottom elevation of the top column piece which will equal the top elevation of the first piece plus the joint distance between spliced columns, usually equal to 2".

According to step (8), (9) and (10) of the column segmentation algorithm (Figure 4.3), top and bottom elevations of the bottom and top column pieces are calculated relative to the top elevation of the closest intersecting spandrel or beam to the middle of their parent column. Since between each segmented column and its closest intersecting beam/spandrel an aggregation relationship is built (Figure 4.15, 4.16 and 4.18), the top elevation of closest intersecting beam/spandrel to the parent column of top and bottom column pieces can be tracked. Through this then top and bottom elevations of the bottom and top column pieces are derived and added to their properties

- With their top and bottom elevation available, height of the newly created column pieces is calculated. With their height provided and since the cross section profile of new columns is the same as the parent column, volume of the top and bottom columns are then calculated, which are essential inputs for quantity take off, cost estimation and other preconstruction activities.

4.10 Conclusion and Next Steps

This chapter discussed a four step framework to perform an automatic semantic enrichment of design models for the purpose of evaluating and preparing model objects for preconstruction activities namely QTO and CE. First a product information model was developed, followed by a problem solving algorithm to infer new information needed to perform a BIM-based QTO and CE activities. Then a set of rules were designed to implement the designed algorithm. It was shown with the test models how the inferred facts were added to design models and users could inquire about them.

In the next steps algorithms and rule sets are developed for evaluation and preparation of other major product types. Also the process will be taken one step further and a knowledge-based system framework will be developed to use the semantic enrichment for automatic design, predicting design features absent from models and adding them to models.

This framework can be adopted for other building systems to fill the gap between available and required BIM design information in different project stages. Yet, considering that SEEBIM uses a simplified geometry of objects and does not deal with curved shapes, there are some inherent limitations to using the rule sets on complex models with free form objects. Future efforts can expand the rule sets and required operators to handle various geometric forms.

CHAPTER V

A KNOWLEDGE BASED SYSTEM FRAMEWORK FOR AUTOMATIC EVALUATION AND PREPARATION OF BIM-BASED DESIGN FOR CONSTRUCTION

5.1 Introduction

The AEC industry has been on a fast track in adopting BIM in recent years; yet the BIM adoption for many preconstruction and construction activities has been slow due to large gaps between BIM-assisted design information and required preconstruction and construction information. Moreover, the current workflow for adoption of BIM in activities like quantity take-off (QTO) and cost estimation (CE) is not cost-effective and practical especially for small and medium sized companies.

A knowledge-based system (KBS) framework is designed to represent the acquired knowledge of construction experts, infer the knowledge required to perform QTO and CE activities and produce the results in the form of enriched IFC design models and tabular QTO information.

This framework is deemed to streamline flow of information from BIM design platforms to preconstruction activities in the AEC project. It acquires the knowledge of construction people and not only provides the necessary design information to construction people but also makes it accessible to design people. Implementation of this framework facilitates adoption of BIM for contractors and sub-contractors specially small and medium sized companies with less resources to implement BIM using the current workflow. It also

attempts to make the adoption of BIM cost-effective for QTO and CE activities in which construction parties are involved in fast track processes and limited time and resources which often makes building the new models with required information and appropriate object representation impractical.

A prototypical solution was developed to automatically modularize monolithic precast concrete slabs and provide their quantitative information to construction users for CE, bidding and production planning purposes.

5.2 Knowledge-Based Systems Overview

Knowledge-Based Systems (KBS) provide a platform to acquire, represent and process data, information and knowledge to generate new knowledge. Unlike traditional information systems they can act as decision makers and serve like an expert on demand [54, 55]. KBS have emerged from the Artificial Intelligence (AI) field and are employed for numerous purposes in various industries.

Knowledge in the sense that is used in KBS can be defined as a system, that provides the ability to manipulate, transform or create data and information to make a decision, perform skillfully or solve a problem [56]. One useful classification of knowledge that grasps two of its important dimensions is (i) conceptual knowledge that is “understanding of the principles that govern the domain and of the interrelations between pieces of knowledge in a domain” versus procedural knowledge defined as “action sequences for solving problems” [57]; (ii) explicit knowledge that involves articulated and structured or semi-structured knowledge versus tacit knowledge built by experience, guided by intuition and residing in one’s subconscious [58].

A closely related concept to KBS is Knowledge Based Engineering (KBE). KBE has mostly been classified as a special type of KBS. Cooper & La Rocca [60] defined KBE as “the use of dedicated software language tools (i.e. KBE systems) in order to capture and re-use product and process engineering knowledge in a convenient and maintainable fashion.” The ultimate objective of KBE is to reduce the time and cost of product development by automating repetitive, non-creative design tasks and by supporting multidisciplinary integration in the conceptual phase of the design process and beyond.

The notion that identifies automation of repetitive and non-creative design tasks as one of the major benefits of implementing KBE systems, is shared by many researchers. This concept, highlights the fact that by significant time and cost savings resulted from automation of repetitive tasks, designers can focus more of their efforts on creative aspects of design [61, 94].

5.2.1 Knowledge Based Systems Architecture

The two major components of the KBS architecture include a knowledge base and a reasoning engine [54]. Some researchers have also included a task [74, 75] and a user interface layer [76, 77] as essential and separate components of a KBS structure. Figure 5.1 illustrates structure of a knowledge-based system.

Domain layer consists of a knowledge base that serves as a repository that represents the knowledge acquired from various domains and represented using different representation tools. Knowledge acquisition and representation deal with content and format of knowledge respectively and enhance availability and usability of knowledge [76]. Various textual, graphical and computer-interpretable knowledge representation conventions and tools have been developed to standardize knowledge modeling in different

domains. A knowledge base represents the acquired domain knowledge using an ontology. User interface systems enable interactions of KBSs with users [76]. For efficient communication, these interactions should consist of two main aspects of (a) receiving inputs from users that outline users' organization preferences, limitations or requirements.

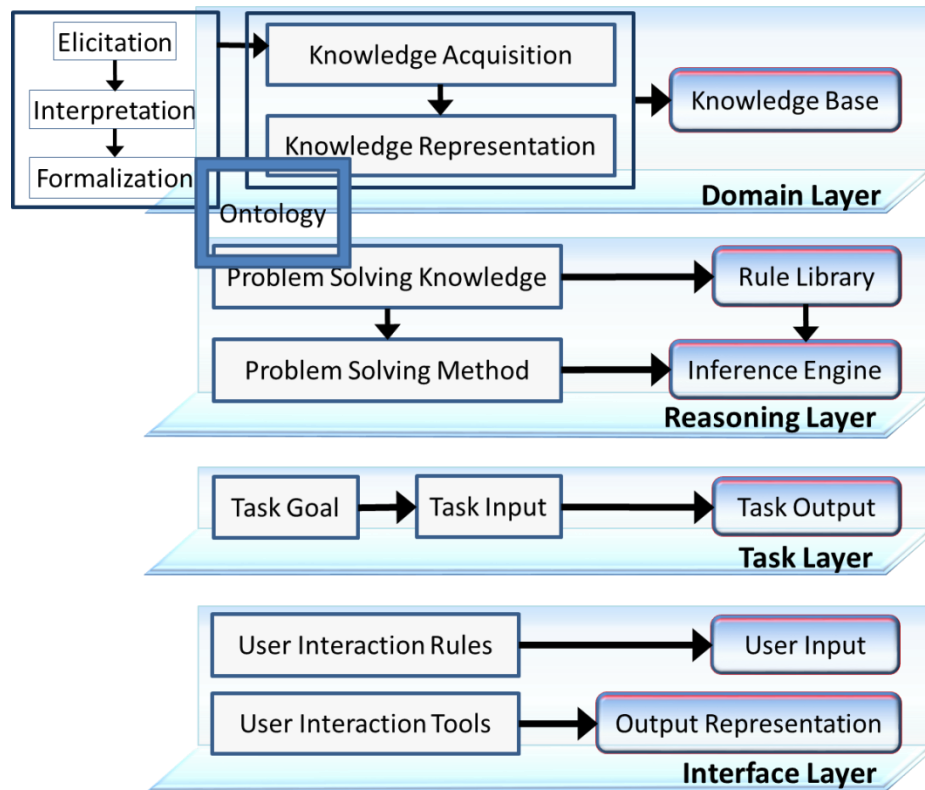


Figure 5.1: Knowledge based systems structure

These inputs are used during the reasoning process to refine problem-solving strategies and achieve a dynamic and customized solution based on users' needs; (b) representing the outputs of reasoning and task layer based on users' criteria for selecting, filtering and grouping outputs.

5.2.2 Proposed KBS Framework for Preconstruction Activities

The reviewed knowledge-based systems assume that design models used include all the information required for cost estimation about designed products and their features

and that the representation method and units in design models fit the cost units of manufacturers, fabricators and constructors. In other words, they only extract information represented explicitly in design models, but do not modify the design to reflect the fabrication and installation units critical for cost estimation. They do not anticipate product features missing from design in earlier stages of a project, nor attempt to enhance and complete the information retrieved from design models to fill the gap between design model data and data required for a project' cost estimation. Hence, most designed systems would only work under ideal situations when late project information is available early in the project for design entities and are contributed to design and represented in design models, which is rarely the case.

The proposed KBS framework, depicted in Figure 5.2, aims to build on the previously developed KBS frameworks and modify and improve them to depict real project work limitations. This is achieved by designing a framework to adjust design models and make them suitable for cost estimation without the need for rebuilding the design models. The key extension is to infer the knowledge critical for accurate cost estimation about missing design features. Thereby the proposed system attempts to enhance the knowledge extracted from design models and to automate the current mostly manual and time-consuming QTO and CE process.

A KBS framework was developed to provide a streamlined, 3D parametric model based quantity take off and cost estimation for construction products. This framework is represented in Figure 6 and includes the 4 layers of domain, reasoning, task and interface, designed for the precast concrete products which comprises the area chosen to implement a proof of concept for this research effort.

Several precast companies have collaborated and provided their company standards, practice manuals and their historical project cost estimation information. The principal researcher of this effort co-located for a few weeks with the industry representatives to acquire domain knowledge of various experts including estimators, structural engineers, plant managers and erectors.

Process maps for each product type was developed that identify the type of information required for features and functions composing a product. In the product information process maps, modules of rule sets required for each feature and function were identified. To develop the rule sets, various design conditions were devised and presented to experts and the design process adopted by experts were traced and recorded. This process resulted in decision trees that represent the rules used in the decision-making process.

The developed process maps and decision-trees were reviewed by experts representing different segments of the industry to identify other potential decision paths. The knowledge acquisition process, of course, is a repetitive cycle where each cycle involves modification- expansion, deletion and change - of previously developed decision-trees and process models and development of new ones.

5.2.3 Cost estimation Methods: Adoption in the Framework

A combination of activity- and feature-based product decomposition is used in this research. The study investigates a variety of design features that compose a specific product type, the supply chain process and activities that are required to fabricate each feature, and identifies design variables that affect cost of each activity and therefore are important to be provided for cost estimators.

The main goal of the reviewed CE methods has been defining relationships of different design variables to cost of a project using historical data and applying various machine learning and optimization methods [11]. The focus in building the knowledge base

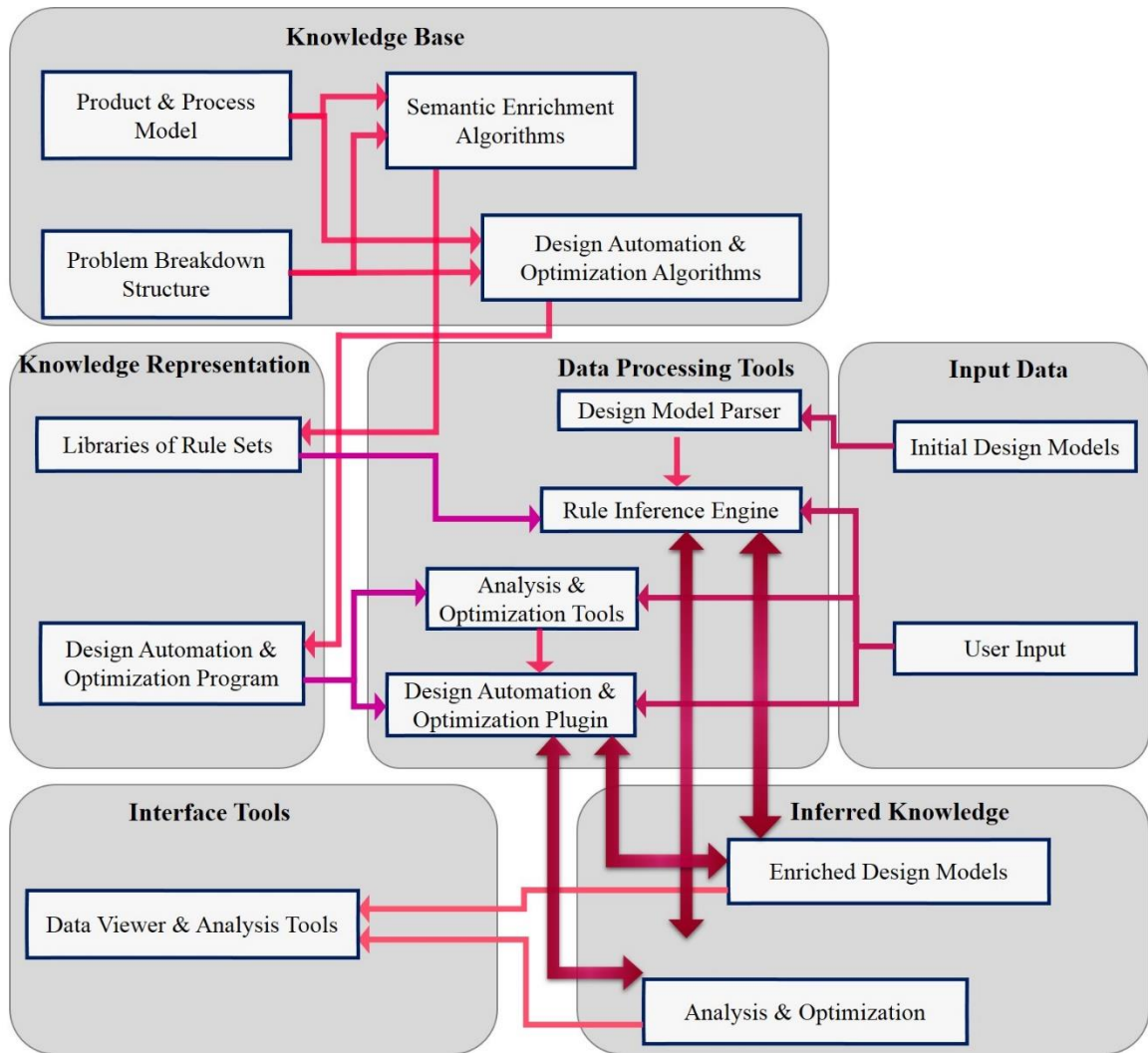


Figure 5.2: The proposed KBS framework

of this framework is not to define cost relationships, rather to identify existence of those relationships between different variables and cost of a project and providing value of these variables to users, when they are not readily available in design models, through building

a rule library and a rule processing engine. When the value of different variables are determined and provided to users they can then plug them in their formulas that are built based on their production process and local economic conditions.

5.3 Problem Definition

Precast concrete slabs designed and modeled by architects and structural engineers working for design parties are often monolithic pieces passing through several column bays. These huge monolithic elements cannot be fabricated and erected. They need to be segmented to narrower pieces usually with a maximum width of 15', based on our interviews with several precast concrete companies.

Providing the information about the size and geometry of each slab piece is critical for construction companies and is a prerequisite for detailed design of slabs, production, shipping and erection planning as well as quantity take-off (QTO) and cost estimation (CE). Many of the cost contributing factors like building forms, number of concrete pours, number of slab connections with other building elements, amount of reinforcement required in slab design and weight of each slab piece that determine the required number of truck loads and appropriate crane type and capacity for the project, depend on the size and geometry of each slab piece.

The information that can be extracted by the design models that are provided by design entities to construction parties is limited to the total volume and weight of concrete used in slabs which is not enough for the decisions made throughout the supply chain or providing an accurate QTO and CE by precast concrete companies.

The same modularization problem and lack of necessary information in design models exist for columns, discussed in chapter 4, and wall panels. Thus, those precast concrete companies that decide to adopt BIM, have to develop their own models. Only a fraction of the projects for which companies prepare QTO and CE and participate in their bidding process is awarded to each company. Considering that, most companies can't afford the resources required to build a BIM model for each project that they bid for.

Moreover, the manual evaluation of design and modularization of slabs is error-prone and difficult to optimize. In the manual modularization process estimators have to rely on their judgment and prior experience which are subjective and difficult to formalize, communicate and standardize. When errors occur and less than optimal element design is used for the rest of the supply chain, they lead to additional costs to the company that won the bid and project delays. Even when the initial design during the QTO and CE is modified and improved during the detailed design, since companies' compensation is based on their initial bids, the cost change has to be absorbed by the construction company which can lead to less project profitability.

In addition to automating the process and saving time and providing the potential for less errors and more optimal design solutions, when the QTO and CE activities are performed using BIM and their quality is improved the results can be used as the basis for later detailed design, production planning and fabrication. Currently and based on our field studies, there is a disconnect between the QTO and CE and subsequent activities after winning the project, which creates considerable waste and rework in the process. BIM based QTO and CE using the proposed KBS framework for semantic enrichment of designs and automated modularization and preparation of models can help not only streamline flow

of the information from design phases to QTO and CE but also from them to other preconstruction and off-site and on-site construction activities.

5.4 The KBS Framework Implementation: Solution Overview

The steps of implementing the developed KBS framework as well as the software products used during the implementation are illustrated in Figure 5.3. The knowledge base development for precast concrete slab segmentation was executed with a focus on Double-Tee (DT) slabs that comprise the majority of slabs used in the precast concrete industry.

The whole process can be divided into two major steps:

- (1) *Semantic enrichment of the initial design models.* Span of the slabs and column bay lengths of bays that each slab is passing through are inferred from the model and added to the design model
- (2) *Automatic optimized design of slab pieces.*
 - (i) Structural analysis to find maximum structurally feasible width for various loading conditions and slabs of different span lengths
 - (ii) Developing a solution for automation and optimization of slab piece design. As illustrated in Figure 5.3, this plugin received the data input from various sources including enriched design model data, results of the structural analysis as well as the user input that reflected the company preferences and limitations. Then the data input from these sources were integrated and processed in an algorithm developed to suggest the best slab modularization scheme for various possible design situations to meet the predefined criteria by users. The results are presented in two ways:

- The slab pieces and their width, length and elevation information were added to the enriched design model.
- The quantity of slabs of various sizes along with other QTO information was written in Excel tables.

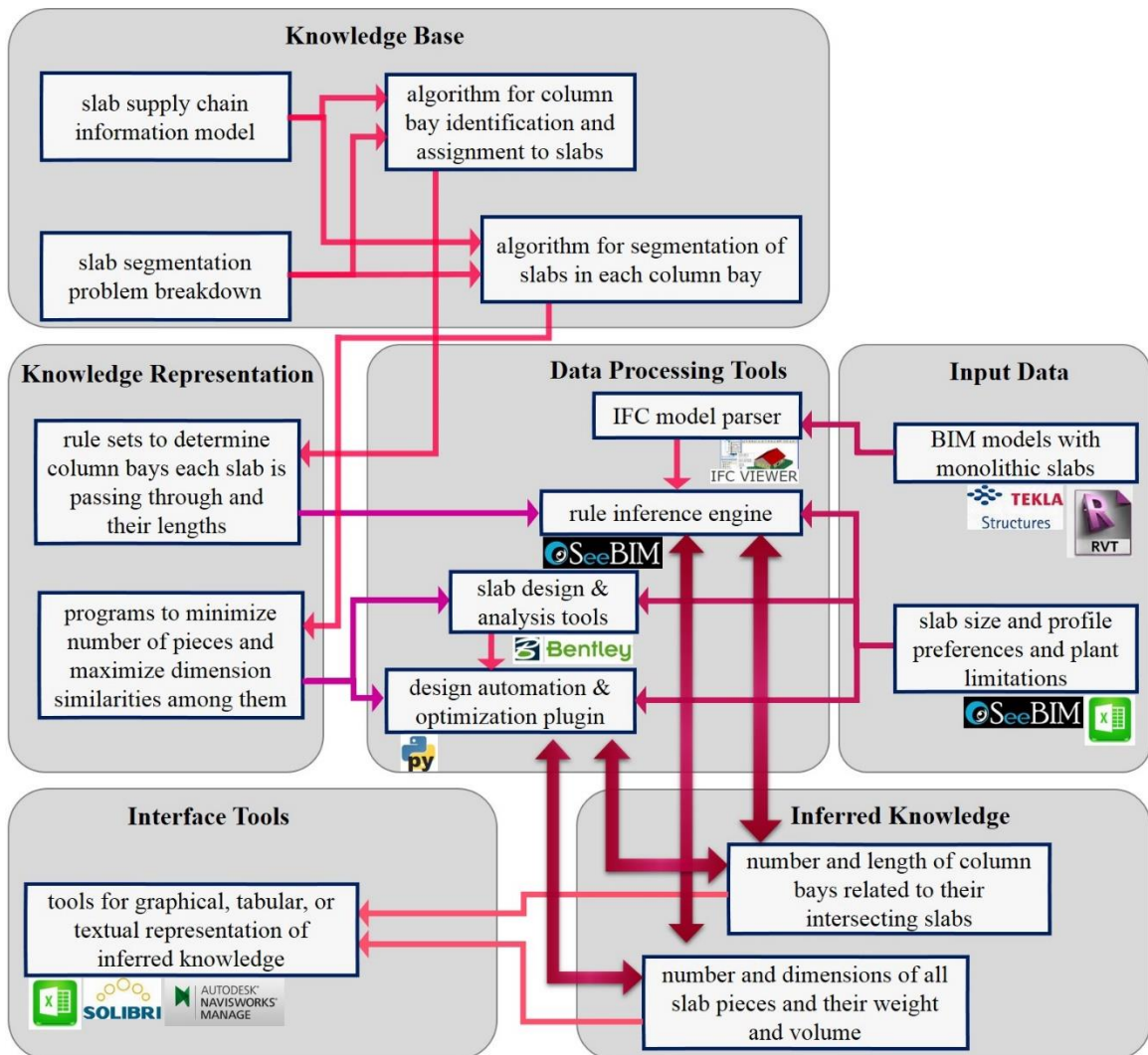


Figure 5.3: Implementation of the KBS framework for precast concrete slab segmentation and quantity take-off

First the product information model throughout the slab design, fabrication and erection was developed. Product models and an example developed for precast concrete columns

were provided in Chapter 4. Then algorithms for each of the above two steps were developed.

5.5 Semantic Enrichment of Design Models

The goal here is to infer the information about column bays and their lengths for each slab that passes through them. A column bay is defined as the horizontal distance parallel to slabs' direction between centroids of two neighboring columns. Providing this information is important since DT slabs are generally segmented in a way that their joints lie in line with the center of intersecting columns. One reason for this is to make sure that DT stems are located in places that they interface with a beam, a spandrel or a wall panel and not with a column. Hence, the distance of column bays provide the first guideline for modularizing slabs.

As mentioned in Chapter 4, SEEBIM was adopted for semantic enrichment in this research work. A categorization of operators and attributes in SEEBIM and their function was discussed in detail in that chapter. Figure 5.4 shows all the object attributes and operators of different categories used for slab modularization. The operators in boxes color-coded with purple headings are used in the IF clause of the rules to analyze the object attributes and spatial and non-spatial relationships of objects. Those in boxes color-coded with blue headings are used in the THEN clause of rules to perform a task and add the inferred semantics to BIM-based design models. These attributes and operators were used in a set of rules to implement the algorithm depicted in Figure 5.5.

5.5.1 Developed Rule Set for the Slab Modularization

The following provides a summary of the rule set developed for semantic enrichment of models for slab modularization purpose:

Slab Classification. This is the first step in semantic enrichment of models for slab modularization. Classification rules, and examples of them were discussed in detail in

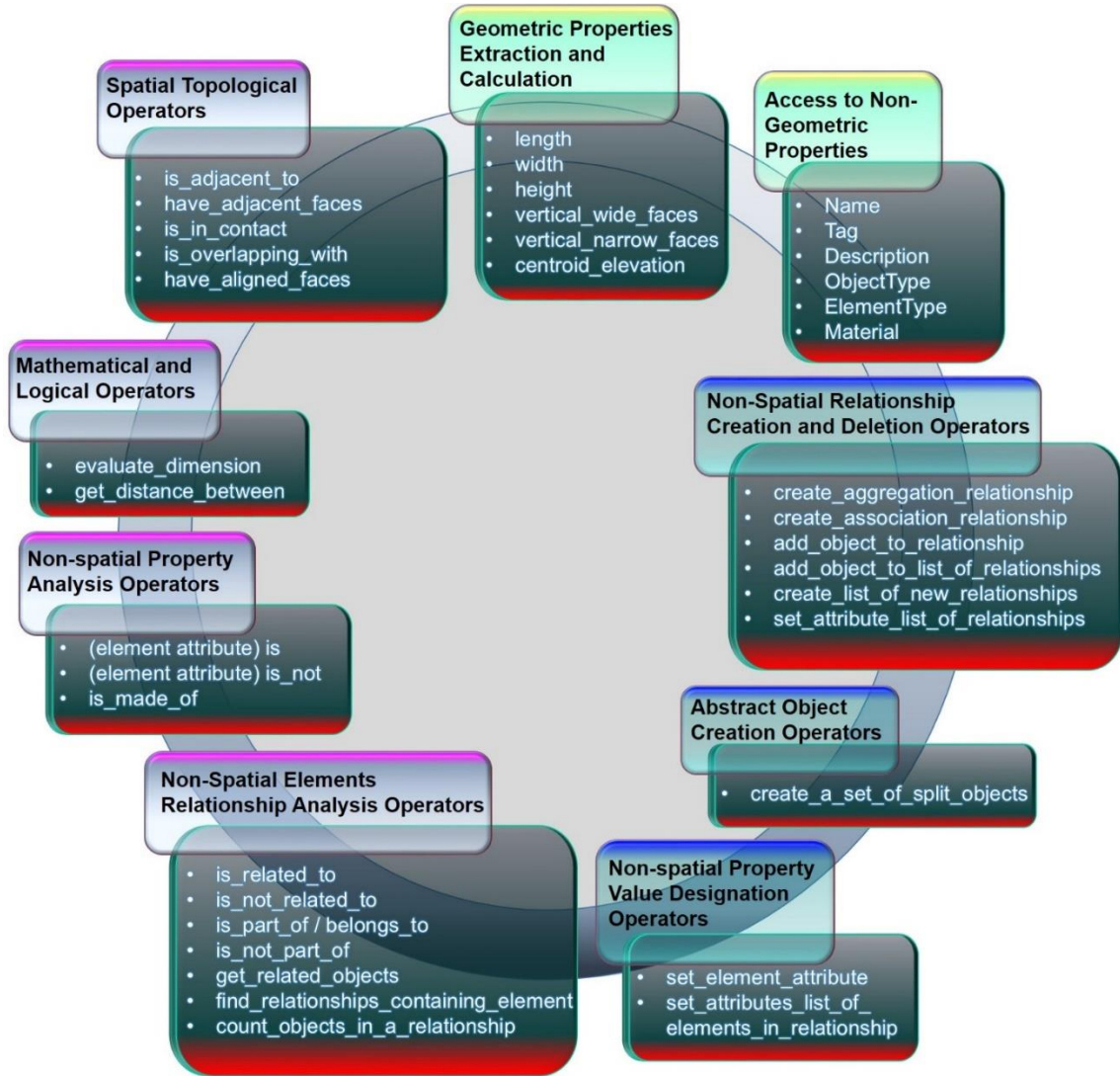


Figure 5.4: List and categorization of object attributes and operators used in semantic enrichment of models for slab modularization

Chapter 4. Like beams, slabs also can be classified based on their profile geometry. As shown in Figure 5.6, small overlap between hollow-core and flat slab, the profile height

range for the three slab types are distinct. For the purpose of identifying DT slabs profile height condition was used.

Slab Width Assignment. Moreover the information about object width and length is implicit in IFC models. SEEBIM is able to access, infer and work with various attributes

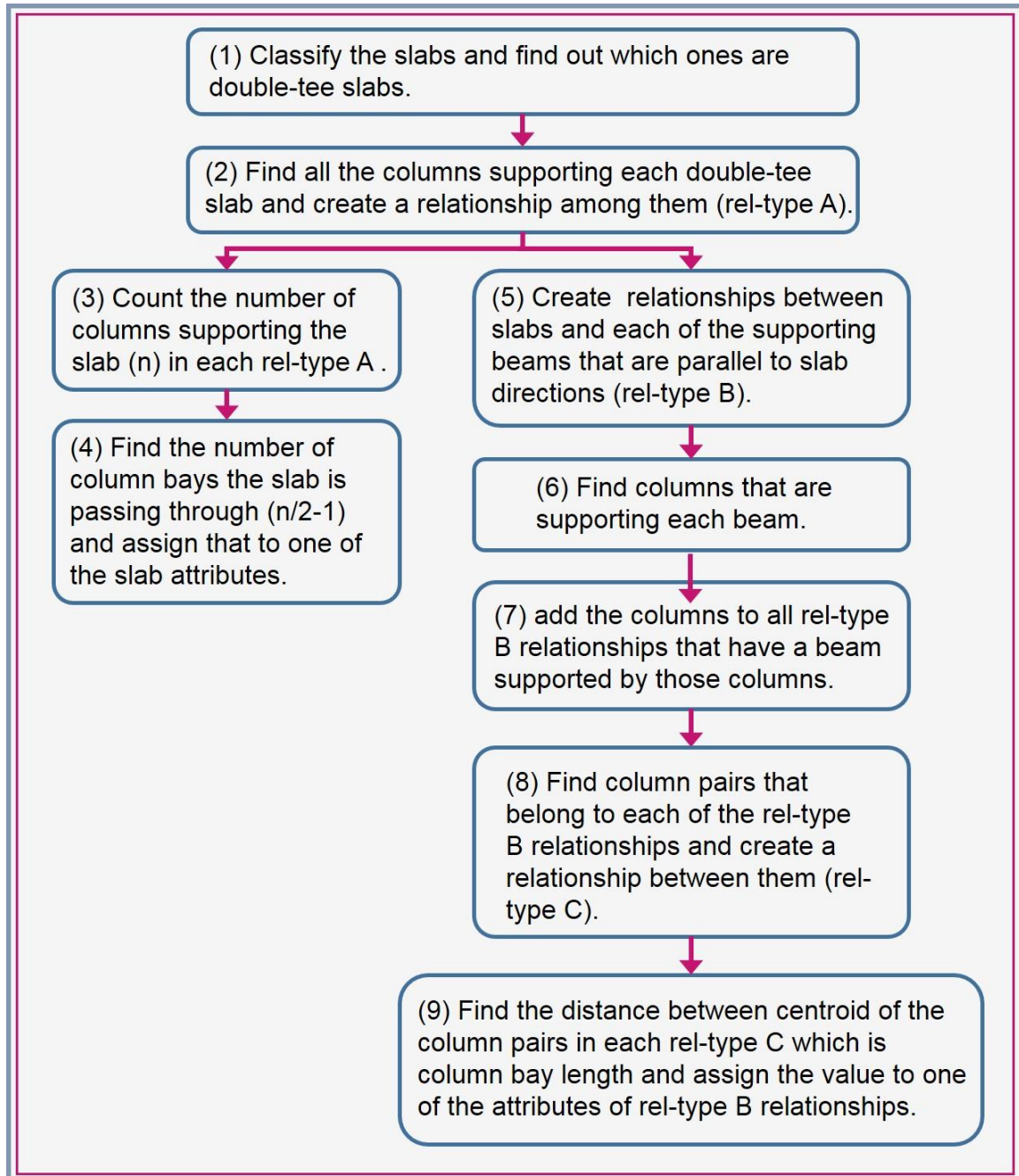


Figure 5.5: The algorithm developed for developing the rule set required for semantic enrichment for slab modularization

of objects including width. Yet for the purpose of using it for automated design, it was required to provide the width value explicitly for each slab in one of its IFC attributes. Slab width was assigned to each slab's *ObjectType*.

Assignment of Number of Bays Each Slab is Passing Through. Step 2-4 of the algorithm depicted in Figure 5.5 explain the process for finding and assigning the number of bays each slab is passing through. First a relationship between each slab and all the intersecting columns is created which is referred to as rel-type A in Figure 5.7. As depicted in Figure 5.10 that represents selected parts of the resulting enriched IFC file, the *Name* and *Description* value of "columns_supporting_the_slab" is assigned to this relationship. Then another rule finds the "columns_supporting_the_slab" relationship that each slab belongs to as *RelatingObject*. It uses the operator *count_objects_in_a_relationship* to count

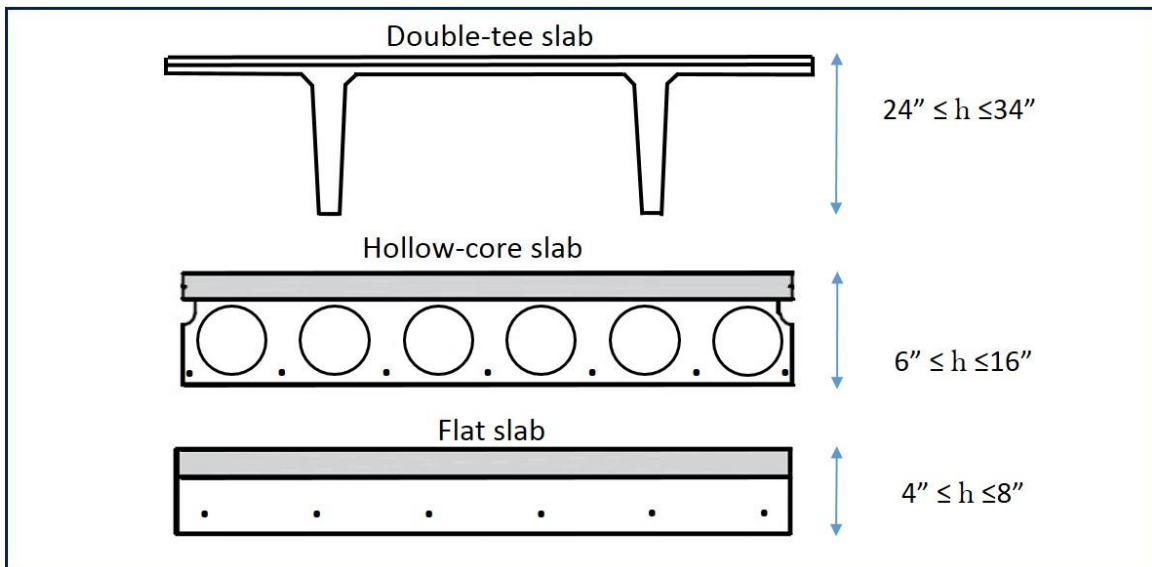


Figure 5.6: Height range of different types of precast concrete slabs used for slab classification

the number of *RelatedObjects* in this relationship. These *RelatedObjects* are the columns that support each slab. The rule then using the *change_element_attribute* operator changes

the *Description* of slabs in each of those relationships according to the formula $\frac{n}{2} - 1$ in step 4 of the algorithm in Figure 5.5. For example, as you can see in Figure 5.7 each slab is supported by 8 columns but is passing through 3 bays.

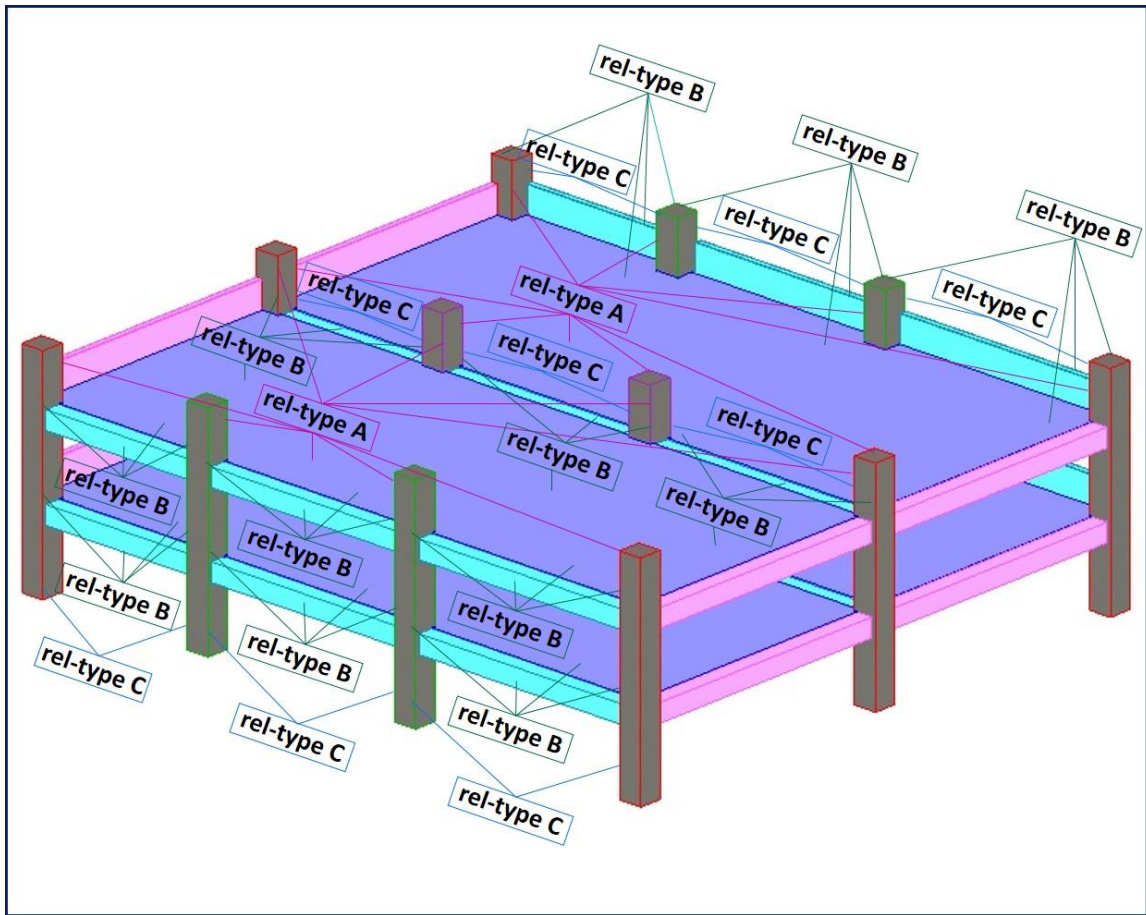


Figure 5.7: Relationships created between model objects for identifying column bays and assigning them to slabs

Finding Column Bay Lengths and Assigning Slabs to Column Bays. From all the possible column-column relationships that can be built among the columns supporting each slab, we needed to only select those column pairs that (i) are neighbor of each other; and (ii) the axis between their centroid is parallel to the intersecting slab's direction and not

perpendicular to it. This means selecting column pairs at the end of blue beams and spandrels in Figure 5.7 and not those at the end of pink beams and spandrels.

First as explained in step 5 of the algorithm in Figure 5.5, beams and spandrels that (i) their *vertical_wide_faces* are adjacent to *vertical_wide_faces* of each slab; or (ii) are overlapping with the slab and their *vertical_wide_faces* are aligned with *vertical_wide_faces* of the slab, are selected. This means all the blue beams and spandrels in Figure 5.7. Then a relationship named “*column_pair_and_beam_supporting_the_slab*” with *Description* and *domain_type* of *column_bay* is created between slabs and each of those beams and spandrels. This relationship is depicted by rel-type B in Figure 5.7.

The next rule, depicted in Figure 5.8, implements step 6-7 of the algorithm and picks up the columns that are adjacent to or overlapping with each beam. By using the

```

IF
<object1_ObjectType> is 'lfcColumn' AND
<object2_ElementType> is 'lfcBeam' AND
(<object1> is_adjacent_to <object2>, 'tolerance', '0.1' OR
<object1> is_overlapping_with <object2>) AND
<object2> is_part_of_relationship_of 'domain_type', 'column_bay' AND
<object2> is_not_related_to <object1> relationship_of 'domain_type',
'column_bay' AND
find_relationships_containing_element relationship_of 'domain_type',
'column_bay' containing <object2> in the list of 'RelatedObjects'

THEN

add_object_to_list_of_relationships <object1>, add to 'RelatedObjects'
relationships found by find_relationships_containing_element

```

Figure 5.8: The rule to build *column_bay* relationships

is_part_of operator, it narrows down the pool of selected beams to those that participate in the *column_bay* relationship. Next line acts as a no-loop control declaration, explained in

detail in 4.8.1.2 section, to make sure that the rule is performed only once on each beam-column pair and avoid an infinite loop. Next, access to *column_bay* relationship that the selected beam is involved in needs to be provided. The selected beams might be part of several different relationships. The *find_relationships_containing_element* operator finds only those relationships with *column_bay* as their *domain_type*. In the THEN clause the column is added to the previously found *column_bay* relationships.

Now in each rel-type B relationship two columns and one beam participate as *RelatedObjects* and one slab as the *RelatingObject*. Note that since each column might pass through several floors, and also in each floor interior columns support two slabs, each column participates in multiple rel-type B relationships. Also since slabs are adjacent to or overlapping with several beams, each slab participates in multiple rel-type B relationships. The last rule, depicted in Figure 5.9, in this rule set implements step 8-9 and finds lengths of the column bays and assigns them to the *column_bay* relationships that slabs are participating in. The rule selects the pairs of columns that belong to the same *column_bay* relationships which means they are neighbor of each other and the distance between their centroids is the bay length. Next it finds all the *column_bay* relationships that the selected column pair participate in. In the THEN clause it first finds the distance between centroid of column pairs and saves it in a variable called *C1_C2_bay*. Then the *set_attribute_list_of_relationships* sets the *Description* of all the relationships found in the IF clause to the *C1_C2_bay* value. The last segment in the THEN clause as well as the line with the *is_not_related_to* operator in the IF clause act as a no-loop control declaration: Each time the rule is executed a relationship (rel-type C in Figure 5.7) is created in the THEN clause and next time if the same column pair is selected by the rule, the

is_not_related_to will not allow the rule to execute again, since already a relationship between those columns was built.

Now as illustrated in Figure 5.10, each slab is in a series of relationships with *domain_type* of *column_bay* and *Name* of “*column_pair_and_beam_supporting_the_slab*”. The *Description* of each relationship is the bay length of those columns that participate in that relationship. Since *domain_type* is an object attribute used internally by SEEBIM to refer to relationships but is not an IFC object attribute, the *domain_type* value (*column_bay*) is not among the written attributes of the relationships in the IFC enriched file.

```
IF
<object1_ObjectType> is 'IfcColumn' AND
<object2_ElementType> is 'IfcColumn' AND
<object2> is_related_to <object1> relationship of 'domain_type',
'column_bay' AND
<object2> is_not_related_to <object1> relationship of 'domain_type',
'checked_for_column_bay_length' AND
find_relationships_containing_element relationship of 'domain_type',
'column_bay' containing <object1> in the list of 'RelatedObjects'

THEN

get_distance_between <object1_centroid_elevation>,
<object2_centroid_elevation> refer to as 'C1_C2_bay',
set_attribute_list_of_relationships 'Description' set to 'C1_C2_bay' + 'bay length'
create_list_of_new_relationships 'IfcRelAssociatesClassification', between
<object1>, <object2>, 'Name', 'Description', 'domain_type'
'checked_for_column_bay_length'
```

Figure 5.9: The rule to add bay lengths to *column_bay* relationships



Figure 5.10: Collection of snapshots from an enriched IFC P21 file created by execution of the slab modularization rule set

5.6 Automatic Design of Slab Pieces

5.6.1 Structural Analysis to Find the Maximum Structurally Feasible Width of Slabs

The next step was to create a library that provides the maximum feasible double-tee (DT) slab width under various possible design conditions. Creating such a library and linking it to design automation and optimization plugin will help users to pull and reuse the analysis results for various projects. Table 5.1 shows the user input for performing the structural analysis and Table 5.2 demonstrates part of the library created for the selected pretopped DT profile with 4" thick flange and 30" depth. The depicted segment in Table 5.1 is for DT span of 50' and 70' which are the DT spans in the test case provided in this chapter. In the future and when there are data interfaces built between structural design and analysis tool and knowledge-based systems used for automated design, the analysis results can be pulled from structural analysis tools in real time. PCI Design Handbook [86] provides all the standard DT profiles used in the industry. Each precast concrete company uses a handful of those standard profiles. Hence, building such a library using structural analysis tools like Bentley LEAP PRESTO, which is used in our analysis, is a practical solution.

Table 5.1: User input for precast concrete double-tee slab structural analysis to find max. DT width and stem reinforcement design in various loading and span conditions

Design code	ACI 318-05
Selected DT profile [86]	pretopped, 4" thick flange, 30" deep
Precast concrete strength	6000 psi
Concrete in flange topping	5000psi
Slab self-weight [86]	based on normal weight concrete 150 pcf
Prestressed strand type & size	9/16" dia. 270ksi, parallel pull
Longitudinal rebar size	#5 or #6 of grade 60
Loading eccentricity	0

To create the library, the structural analysis to determine (i) the total final stress (psi) on the bottom of DTs under prestress plus all dead loads (DL) and all live loads (LL), (ii) ultimate strength (pM_n/M_u), (iii) camber and deflection (inch) under live loads, and for DTs with width greater than 9' also (iv) transverse bending on flanges, all based on the ACI 318-05 design code was performed. The library is created for five live loads ranging from 40-250 psf.

The LL range and values are selected based on the ASCE/SEI 7-10 standard [116] that defines minimum design loads for buildings and other structures. The selected live loads cover the minimum uniformly distributed live loads for most of different building types in this standard from schools and libraries to hospitals, office buildings, recreational uses, heavy manufacturing buildings and more.

The library is created for DTs with the span of 40' to 90' with increments of 5' based on the span range provided for standard DT profile design in the PCI design handbook [86]. Since tests showed that in terms of maximum allowed DT width and stem reinforcement design, 5' difference in span doesn't make a meaningful difference specially in the accuracy required for QTO and CE, the spans in between two can be rounded up to the next number. The selected strand and rebar size and type (Table 5.1) are the typical ones used by most companies for DT stem design. The table is developed by first finding the maximum DT slab width and the minimum number of strands (tendons) required in each DT stem for every DT span and LL combination that satisfied the above mentioned design code requirements.

Using the maximum structurally feasible DT width minimizes the number of pieces for each project which is one of the goals of DT piece design by precast concrete companies. However, the building design, specifically bay sizes as well as fabrication, shipping and erection costs often impose design and fabrication of narrower DTs. These factors will be explained in more detail in section 5.6.2.1. To satisfy these situations

Uniformly Distributed Live Loads (psf)	Slab Width & Stem Reinforcement Design	Slab Span (ft)													
		50'					70'								
40	slab width (ft)						7'-13'	14'	7'	8'	9'	10'	11'	12'-13'	14'
	strands & rebars						6	8	8	8S+2R	10	12	12	14	16
60	slab width (ft)						7'-10'	11'-13'	14'	7'	8'	9'	10'	11'	12'
	strands & rebars						6	8	8S+2R	10	12	12S+2R	14	16S+2R	16S+2R+8k C
80	slab width (ft)						7'-12'	13'	14'	7'	8'	9'			
	strands & rebars						6	8	10	12	14	16S+2R			
100	slab width (ft)	7'-9'	10'	11'	12'	13'	14'	7'	8'						
	strands & rebars	8	8S+2R	10	10S+2R	12	14	16	16S+2R+8k C						
250	slab width (ft)	7'	8'												
	strands & rebars	16	16S+2R												

Table 5.2: Results of the slab structural analysis including the max structurally feasible DT width and stem reinforcement design for each DT width

minimum number of strands that satisfy the above mentioned structural requirements for DTs narrower than maximum structurally allowable width and equal or wider than the minimum feasible DT width (7') are also calculated.

Hence, the table provides stem reinforcement design for all possible DT widths and lengths (spans) and loading conditions. In Table 5.1, for example the far left cell in cross section of 80 psf LL and 50' span reads 14' and 10, meaning that the maximum allowed

DT width is 14' and 14' wide DTs in this design situation require 10 strands. In the same design situation, 13' wide DTs require 8 strands and so on.

In a number of situations like the cell in the far left cross section of 70' span and 60 psi live load the numbers under the width reads like “16S+2R+8k C”. This means that this design situation requires 16 strands and 2 rebars and 8000 psi concrete. Rebars are used when the number of strands specified were not satisfying the structural criteria and are added to increase the moment capacity. There are two reasons to opt for this solution instead of adding to the number of strands: Either the depth of stems and the available DT forms don't allow adding to the number of strands (maximum number of strands in most standard DT forms is 8 per stem.) or since per unit of measurement (foot) rebar on average costs less than a strand, if adding 2 rebars was enough to meet the structural requirements, that option was preferred. The 8000 psi concrete was only used when strands and rebars were not enough to meet the structural requirements and without the increased concrete strength, the maximum allowed DT width had to be lowered. These choices were preferred since based on the experience of the interviewed industry experts, the economic benefits achieved by less pieces per project with designing wider DTs surpasses the cost of added rebars and stronger concrete.

5.6.2 Automation and Optimization of Slab Piece Design

The goal in this section is to develop and implement an algorithm that suggests the best feasible slab modularization scheme that meet the design limitations and the user criteria. The implemented algorithm would work like a plugin that receives, integrates and processes the information from the enriched IFC models (Table 5.5), the structural analysis

results (Table 5.2 and Table 5.3), and the user input that reflects the company preferences and limitations (Table 5.4).

5.6.2.1 User Input: Company Preferences and Limitations

Preferred DT Width. Due to various factors affecting the supply chain of double-tees (DTs) and additional shipping and handling costs incurred by companies for wide DTs, many times the ideal DT slab width for companies to minimize the fabrication, shipping and erection costs of DTs is different than the maximum structurally allowed width. An example of these factors that affect the total cost for companies is the permit fee in most U.S. states for shipping DTs wider than 12'.

This ideal DT slab width to minimize the total supply chain costs of DTs is here referred to as “Preferred Width” and is denoted by W_{pref} in the algorithm developed for the automated design of DT pieces. The preferred width is determined by companies based on their historical data of prior projects. The preferred width in our interviews with several precast concrete companies is determined to be 12' by most companies in the U.S. Hence, in the test case 12' is used for W_{pref} when running the program. Yet it is recognized that this preferred width might be different for different companies. So W_{pref} is defined as a variable whose value is provided by users and the algorithm optimizes the DT piece design for different values of W_{pref} .

Maximum Feasible DT Width of the Plant. While generally segmenting slabs according to companies' W_{pref} value minimizes the fabrication and erection costs, sometimes and depending on the bay length fabricating wider slabs can help reduce the number of pieces by one, i.e. instead of two narrow slabs one very wide slab is used. In these cases and based on the experience of the precast concrete companies the saving that

results from reducing the number of slabs surpasses the additional cost of shipping and erecting wider slabs. In our designed algorithm, only when designing slabs wider than W_{pref} leads to decreasing the number of slab pieces by one, these slabs are used.

The maximum feasible DT width that can be used for these wide slabs is determined by the forms in each plan and other plant design factors like plant's gate width. Here, this maximum feasible width is denoted by $W_{plant.max}$. Based on our studies while a few companies can produce 15' wide DTs, maximum width for most companies in the US is 14', hence this value is used in the test case.

Minimum Feasible DT Width of the Plant. This is determined based on the DT profiles that each company can fabricate which in turn depends on the installed forms in the plant. The maximum feasible DT width, denoted by $W_{plant.min}$, in each plant needs to be larger than the DT stem centroid to centroid distance. According to the PCI design handbook [86] this distance for standard DT profiles range from 4' to 7'-6". Since this distance in the selected DT profile for the test case was 6', the minimum feasible DT width of 7' was used.

Table 5.3: Results of the design model semantic enrichment and performed structural analysis used as input for automated and optimized slab design

Symbols	User Inputs: Outputs of Semantic Enrichment & Structural Analysis
L_{bay}	Semantic enrichment output: Length of column bays that each slab is passing through
$W_{struc.max}$	Structural analysis output: maximum structurally feasible DT width

Table 5.4: User input reflecting company preference and limitations used as input for automated and optimized slab design

Symbols	User Input: Company Limitations & Preferences	Values Used in the Test Case
W_{pref}	Preferred DT width	12'
$W_{plant.max}$	Maximum feasible DT width of the plant	14'
$W_{plant.min}$	Minimum feasible DT width of the plant	7'

Table 5.5: Semantically enriched IFC test model data extracted to be used as input for automated and optimized slab design

Symbols	Values Extracted from the Initial Test Model & Added to the Enriched Model	
	Slab Type1	Slab Type2
slab span	49.93'	69.5'
# of slabs in the model	3 (level 1,2 &3)	3 (level 1,2 &3)
W_{bay} of bays that each slab is passing through	32', 45', 48', 50', 53'	32', 45', 48', 50', 53'

As explained earlier W_{pref} is the company desired width for DTs and this algorithm is designed to maximize the use of W_{pref} unless $W_{struc.max}$ is lower than W_{pref} and structurally it is not feasible to use W_{pref} in the design. For this reason first the minimum of W_{pref} and $W_{struc.max}$ for each slab in the model, denoted by $Typ (W)$, is found and used in the formulas. Moreover, when designing slabs using $W_{plant.max}$ decreases the number of slab pieces by one, these slabs are used, unless again the $W_{struc.max}$ is smaller than $W_{plant.max}$. Thus, we need to find the minimum of these two variables, denoted by $Max (W)$ and use that in formulas.

$$Typ (W) = \text{Min} (W_{pref}, W_{struc.max})$$

(1)

$$Max (W) = \text{Min} (W_{plant.max}, W_{struc.max})$$

(2)

Since the algorithm intends to use as many slabs with the width of $Typ (W)$ in each bay, the first step is to find out if $Typ (W)$ is a divisor of L_{bay} :

$$RL = L_{bay} \% Typ (W)$$

(3)

where RL is the remainder of dividing L_{bay} by $Typ (W)$. RL stands for “Remaining Length” which indicates that when the bay slab is segmented using slabs of $Typ (W)$ width, the remaining length of bay will be equal to RL . value. Table 5.6 depicts the outputs of formulas used in the rest of the algorithm and their meaning. The formulas provide the number of slab pieces with width of $Typ (W)$ in each bay and the width and number of slabs used in the remaining length of bay.

The rest of the algorithm is structured based on the value of $R.L.$ and can proceed in one of the following four directions:

$$(1) \text{ if } (RL = 0) \tag{4}$$

$$\text{then } DT_{quant} = DT_{quant.typ} = L_{bay} / Typ (W) \tag{5}$$

$$DT_{quant.last} = 0 \tag{6}$$

In this situation the whole bay length will be divided into slab pieces of $Typ (W)$ width.

$$(2) \text{ if } (RL \geq W_{plant.min}) \tag{7}$$

$$\text{then } W_{DT.last} = RL \tag{8}$$

$$DT_{quant} = [L_{bay} / Typ (W)] + 1 \tag{9}$$

$$DT_{quant.typ} = [L_{bay} / Typ (W)] \tag{10}$$

$$DT_{\text{quant.last}} = 1 \quad (11)$$

In the second case, the whole RL will be one slab. This will produce DTs with typical width for most of the bay length and at one end of the bay one narrower DT with a width somewhere between $W_{\text{plant.min}}$ and $\text{Typ}(W)$. DT_{quant} is the floor of L_{bay} divided by $\text{Typ}(W)$ plus one for the DT that goes to the remaining length of the bay.

$$(3) \text{ if } (RL < W_{\text{min}} \ \& \ RL \leq \text{Max}(W) - \text{Typ}(W)) \quad (12)$$

$$\text{then } W_{\text{DT.last}} = L_{\text{bay}} - (\text{Typ}(W) * ([L_{\text{bay}} / \text{Typ}(W)] - 1)) \quad (13)$$

$$DT_{\text{quant}} = [L_{\text{bay}} / \text{Typ}(W)] \quad (14)$$

$$DT_{\text{quant.typ}} = [L_{\text{bay}} / \text{Typ}(W)] - 1 \quad (15)$$

$$DT_{\text{quant.last}} = 1 \quad (16)$$

Formulas in the third situation will produce DTs with typical width for most of the bay length and at one end of the bay one wider DT with a width larger than $\text{Typ}(W)$ and smaller or equal to $\text{Max}(W)$.

$$(4) \text{ if } (RL < W_{\text{plant.min}} \ \& \ RL > \text{Max}(W) - \text{Typ}(W)) \quad (17)$$

$$\text{then } W_{\text{DT.last}} = (L_{\text{bay}} - (\text{Typ}(W) * ([L_{\text{bay}} / \text{Typ}(W)] - 1))) / 2 \quad (18)$$

$$\text{if } W_{\text{DT.last}} \geq W_{\text{plant.min}} \quad (19)$$

$$\text{then } DT_{\text{quant}} = [L_{\text{bay}} / \text{Typ}(W)] + 1 \quad (20)$$

$$DT_{\text{quant.typ}} = [L_{\text{bay}} / \text{Typ}(W)] - 1 \quad (21)$$

$$DT_{\text{quant.last}} = 2 \quad (22)$$

$$\text{if } W_{\text{DT.last}} < W_{\text{plant.min}} \quad (23)$$

$$\text{then } W_{\text{DT.last}} = (L_{\text{bay}} - (\text{Typ}(W) * ([L_{\text{bay}} / \text{Typ}(W)] - 2))) / 3 \quad (24)$$

$$DT_{\text{quant}} = [L_{\text{bay}} / \text{Typ}(W)] + 1 \quad (25)$$

$$DT_{\text{quant.typ}} = [L_{\text{bay}} / \text{Typ}(W)] - 2 \quad (26)$$

$$DT_{\text{quant.last}} = 3 \quad (27)$$

Fourth situation will produce DTs with typical width for most of the bay length and at one end of the bay 2 or 3 narrower DTs. Consider the L_{bay} of 53' with $\text{Typ}(W)$ of 10'. Based on line (18), the $W_{\text{DT.last}}$ is 6.5' which is less than $W_{\text{plant.min}}$. Thus, $W_{\text{DT.last}}$ will be calculated based on line (24) which results in $W_{\text{DT.last}} \approx 7.66'$. Hence, this bay will be segmented to two 10' wide DTs and three 7.66' wide DTs. If the resulting $W_{\text{DT.last}}$ in line (24) is still smaller than $W_{\text{plant.min}}$ then the algorithm will continue with the same logic shown above until the $W_{\text{DT.last}} \geq W_{\text{plant.min}}$, however, that situation rarely occurs.

5.7 Test Case Results

The test model used to illustrate the system results is a 3 floor building structure with two large monolithic precast concrete slabs in each floor that each pass through five bays (Table 5.5). The lengths of the bays were chosen in a way that all the four situations explained in the algorithm above occur.

The code is written in Python but any programming language for which plugins are written to be able to pull data from IFC files as well as Excel sheets can be used. The code is written in a way at the end it automatically writes the results of precast concrete slab modularization into the enriched IFC file as well as Excel sheets which is the form that cost estimators usually use for QTO and CE activities. The following is the information that are provided for users in the output files:

Enriched IFC file: Equal to the number of the slab pieces that the algorithm devises for each slab to be segmented into, IfcSlab entities are created and added to the end of IFC file. Examples of these added entities can be seen in Figure 5.11. Slab width, floor level, span length and number of strands and rebars used in its stem design were added to *Name*, *Description*, *ObjectType* and *Tag* attributes of the entities as seen in Figure 5.11.

Excel tables: The results are provided in two tables. In the first table (Table 5.6) the slab piece is organized per floor level. The tables and their information items were designed based on actual QTO tables that were collected from precast concrete companies. The code finds all the slabs with equal width and span length that are in the same floor level and writes their size and concrete and reinforcement quantity information in one row. Quantities were first provided per piece and then total quantity of same size slabs in each level is provided. Total concrete volume was calculated multiplying the DT slab profile area by its span and by number of DT pieces in each floor, where DT slab profile area was

extracted from PCI Handbook [86]. The second table (Table 5.7) the same information was provided for DTs of similar size in the whole project as well as total linear feet, volume and weight of concrete used in the whole project.

Table 5.6: Output of the automatic design of the slab pieces: slab piece lengths and widths in each floor level, total quantity of slabs in each level and size, and slab stem reinforcement design

Floor Level	Top Member Elevation (ft)	Member Length (ft)	Member Width (ft)	# of DT Pieces	Total Linear Feet	Total Concrete Volume (cu.ft)	Total Concrete Weight (pcf)	# of Starnds in each DT Piece	Total # of Starnds	# of Rebars in each DT Piece	Total # of Rebars
2nd FL	10.67	49.93	9	1	49.93	271.84	40776.17	6	6	0	0
3rd FL	21.33	69.5	8	1	69.5	355.22	53283.33	12	12	0	0
1st FL	0	69.5	7.33	3	208.5	1,019.10	50955.08	12	36	0	0
3rd FL	21.33	69.5	7.5	2	139	687.28	51545.83	12	24	0	0
3rd FL	21.33	49.93	12	15	748.95	4,826.57	48265.67	8	120	0	0
2nd FL	10.67	69.5	7.67	3	208.5	1,042.73	52136.58	12	36	0	0
1st FL	0	69.5	10	16	1112	6,424.89	60233.33	14	224	0	0
1st FL	0	49.93	12	15	748.95	4,826.57	48265.67	8	120	0	0
1st FL	0	69.5	7.5	2	139	687.28	51545.83	12	24	0	0
3rd FL	21.33	69.5	10	16	1112	6,424.89	60233.33	14	224	0	0
2nd FL	10.67	49.93	8.5	2	99.86	527.04	39527.92	6	12	0	0
2nd FL	10.67	49.93	14	1	49.93	355.06	53258.67	8	8	2	2
3rd FL	21.33	49.93	9	1	49.93	271.84	40776.17	6	6	0	0
3rd FL	21.33	69.5	7.67	3	208.5	1,042.73	52136.58	12	36	0	0
2nd FL	10.67	69.5	8	1	69.5	355.22	53283.33	12	12	0	0
3rd FL	21.33	69.5	7.33	3	208.5	1,019.10	50955.08	12	36	0	0
3rd FL	21.33	49.93	14	1	49.93	355.06	53258.67	8	8	2	2
2nd FL	10.67	69.5	7.33	3	208.5	1,019.10	50955.08	12	36	0	0
1st FL	0	49.93	14	1	49.93	355.06	53258.67	8	8	2	2
2nd FL	10.67	69.5	10	16	1112	6,424.89	60233.33	14	224	0	0
2nd FL	10.67	69.5	7.5	2	139	687.28	51545.83	12	24	0	0
1st FL	0	49.9	8.5	2	99.8	526.72	39504.17	6	12	0	0
3rd FL	21.33	49.93	8	1	49.93	255.20	38279.67	6	6	0	0
2nd FL	10.67	49.93	12	15	748.95	4,826.57	48265.67	8	120	0	0
1st FL	0	49.93	9	1	49.93	271.84	40776.17	6	6	0	0
1st FL	0	69.5	7.67	3	208.5	1,042.73	52136.58	12	36	0	0
1st FL	0	69.5	8	1	69.5	355.22	53283.33	12	12	0	0
2nd FL	10.67	49.93	8	1	49.93	255.20	38279.67	6	6	0	0
3rd FL	21.33	49.93	8.5	2	99.86	527.04	39527.92	6	12	0	0
1st FL	0	49.93	8	1	49.93	255.20	38279.67	6	6	0	0

Table 5.7: Output of the automatic design of the slab pieces: slab piece quantity and stem reinforcement in each size group in the whole project

DT Size Group	Member Length (ft)	Member Width (ft)	Total # of DT Pieces	Total Linear Feet	Total Concrete Volume (cu.ft)	Total Concrete Weight (pcf)	Total # of Strands	Total # of Rebars
1	69.5	7.33	9	625.50	3057.30	152865.25	108	0
2	69.5	10	48	3336.00	19274.67	180700.00	672	0
3	69.5	7.5	6	417.00	2061.83	154637.50	72	0
4	69.5	7.67	9	625.50	3128.20	156409.75	108	0
5	69.5	8	3	208.50	1065.67	159850.00	36	0
6	49.93	8.5	6	299.58	1581.12	118583.75	36	0
7	49.93	14	3	149.79	1065.17	159776.00	24	6
8	49.93	8	3	149.79	765.59	114839.00	18	0
9	49.93	12	45	2246.85	14479.70	144797.00	360	0
10	49.93	9	3	149.79	815.52	122328.50	18	0

Total in the Project	135	8208	47295	1464787
-----------------------------	-----	------	-------	---------

```

6327 #11807= IFCSLAB('hl4LRe2aFUL0n0VnjBbjcR',#41,'Floor:Precast Concrete Slab -
30" thick & 14.0' wide','3rd FL','slab length 49.93' & number of strands 8 & number
of rebars 2,$,$,'double_tee_slab_piece');
6328 #11810= IFCSLAB('9Zg9YYcAPHIF8oZJ0Xbk5i',#41,'Floor:Precast Concrete Slab -
30" thick & 8.5' wide','3rd FL','slab length 49.93' & number of strands 6,$,$,
'double_tee_slab_piece');
6329 #11811= IFCSLAB('L1Dt2Ltf9twczZ$jffOZc',#41,'Floor:Precast Concrete Slab -
30" thick & 12.0' wide','3rd FL','slab length 49.93' & number of strands 8,$,$,
'double_tee_slab_piece');
6330 #11818= IFCSLAB('83JG46TbaG8puKdXapzfeL',#41,'Floor:Precast Concrete Slab -
30" thick & 7.33' wide','3rd FL','slab length 69.5' & number of strands 12,$,$,
'double_tee_slab_piece');
6331 #11819= IFCSLAB('2NbwKITjHN1DeUbMvZ0Dgx',#41,'Floor:Precast Concrete Slab
- 30" thick & 10.0' wide','3rd FL','slab length 69.5' & number of strands 14,$,$,
'double_tee_slab_piece');
6332 #11821= IFCSLAB('TK4M6j20XnVLpYo17kxyrK',#41,'Floor:Precast Concrete Slab
- 30" thick & 7.67' wide','3rd FL','slab length 69.5' & number of strands 12,$,$,
'double_tee_slab_piece');

```

Figure 5.11: Added slab entities to the enriched IFC model and their attribute information

5.8 Conclusion

In this chapter a knowledge-based system framework for automatic semantic enrichment of design models as well as automatic and optimized design for preconstruction and construction activities. The framework was implemented for precast concrete slab modularization as a proof of concept to illustrate how it can automatically infer the information needed for practitioners to be able to segment monolithic slabs and perform QTO and CE on them. The algorithm was designed in a way that it minimized the number of slab pieces while adhering to user preferences and limitations. It is important to note that the framework and design algorithms can easily be edited by users to provide optimal design solutions for different users.

Ideally and for a completely integrated and automated process, such a KBS platform should be integrated with various design and analysis tools like the structural analysis tool used in the illustrated example to pull the necessary information in real time from various sources of creating knowledge throughout a project. This will need solving the interoperability problems among design, analysis and project management platforms. Also right now users insert their input through separate interfaces: one built for developing rules in SEEBIM and Excel for automated design. Integrating these user interfaces and automating transfer of data between the first and second step of the process will improve the user experience in the future.

CHAPTER VI

AUTOMATED DETAILED DESIGN FOR STREAMLINED APPLICATION OF BIM IN PRECONSTRUCTION ACTIVITIES

6.1 Introduction

The current industry practices for QTO and CE activities are mostly manual, time-consuming and error-prone. This chapter proposes enhancement of the proposed KBS framework to be used for automated detailed design. The area of focus is automated design of connections between precast concrete elements. Various design factors impact the quantity and types of connections used among different building elements. This solution enables practitioners to analyze the building design conditions, infer the quantity and type of appropriate connections for each design situation and automatically add the connections to design models so that cost estimators can extract this information from the design models and use for estimation activities.

6.2 Design Automation

The concept of design automation, to a large extent, has been synonymous to Computer Automated Design which suggests using an automated and computer integrated system to assist with the product and project design. The focus of design automation has mostly been on engineering design. While design automation aims to automate the mundane engineering tasks and to predict the design, in doing so most often it attempts to achieve design optimization, and in some forms to even improve the innovation in design. As such the concepts of “*automated design*”, “*predictive design*” and “*design optimization*” are closely tied together.

6.3 Integration with Design Optimization Methods

Throughout the efforts for automated design a variety of solution search and optimization methods have been utilized. Classical mathematical and probabilistic methods have long been used for design automation. Later and with the emergence of evolutionary computation methods they were increasingly applied to design automation especially to more complex problems with a high number of variables and large search space [32]. Evolutionary algorithms evolve the solution space by conducting iterated and interrelated selection and reproduction processes. Several studies [32, 33, 92] have reviewed, classified and compared the research body on mathematical as well as heuristic models used in the facilities design.

The level of design progress determines the amount of available information and how well the problem and its constituent parameters can be defined. Generally formal mathematical methods perform well in the detailed design stages and on the well-formed problems. On the other hand the major advantage of evolutionary methods is in dealing with solving problems with fuzzy objectives and vague structure [33]. While the solutions for detailed design problems are only selected based on quantitative criteria, the conceptual design solutions are selected using a combination of qualitative and quantitative criteria. In the context of cost estimation applications, the underlying methodologies, differences, advantages and disadvantages of mathematical and heuristic methods were presented in Chapter 2, section 2.1 of this work.

Kicinger et al. [33] categorized the structural design optimization efforts in three major groups:

- Topology and layout optimization which is the focus of conceptual design stage
- Shape optimization performed during design development
- Sizing optimization focused on optimizing member profiles and dimensions and performed during the detailed design

To a large extent the same categorization can be applied to the architectural design optimization problems. The design stage determines the amount of available information and how well the problem and its constituent parameters can be defined. Generally formal mathematical methods perform well in detailed design stages and on the well-formed problems. On the other hand the major advantage of evolutionary methods is in dealing with solving problems with fuzzy objectives and vague structure. While the solutions for detailed design problems are only selected based on quantitative criteria, the conceptual design solutions are selected using a combination of qualitative and quantitative criteria.

Examples of research efforts in these three categories using evolutionary algorithms include optimization of topology of truss structures using Genetic Algorithms (GAs) [34, 35], optimization of beam and slab layout using a GA [36], design optimization of tall reinforced concrete buildings using fuzzy logic [37], structural shape optimization [38], truss size optimization by heuristic method of harmony search [39], and optimizing the size of large steel structures [40], and finally optimizing design of the reinforced concrete frames based on cost, constructability, environmental impact, and safety performance of the design using multi-objective simulated annealing [41].

Examples in the architectural design domain include applying evolutionary algorithms to produce novel space compositions during the architectural design [42], using

ant colony method to optimize design of the building envelopes based on lighting and cost performance [43].

One important application domain of design automation has been to explore a large variety of design solutions and to generate distinctive designs. Evolutionary design process is one of the major automated design generation tools that is used mainly during the conceptual design when some of the objectives are unquantifiable and subjective [44]. Shape grammars were combined with evolutionary design computation to generate a shelter design [45]. Maher et al. [46] introduced co-evolutionary design process in which both design requirements and solutions evolved and fitness function changed based on the interactions among requirements and solutions.

Another classical example of design automation problem is layout design optimization. A computer-aided automated design procedure [47] was developed to explore spatial and structural design interactions and to automatically generate spatial designs, building zones, structural system for each zone and room positioning within constraints of the selected structural design. Numerous research efforts chronicled in [48] have focused on automating and optimizing layout design of buildings specially manufacturing facilities. Another study [49] generated developed a set of geometric constraints and objectives to generate and optimize architectural layout designs of residential blocks.

6.4 Integration with Knowledge-Based Systems and Object-Oriented Modeling

The focus of the above systems has been on automatic search for solutions and improving the performance of the solutions against a defined function. Another words, they have

tackled automation of data processing while the processes for acquiring design data, structuring, representation and reusing the data for a broader problem spectrum to a large extent was left unsolved.

Realizing the need for providing flexible and intelligent product design definition and manipulation to cover broader design automation problems more efficiently resulted in using object-oriented parametric design modeling tools in design automation efforts. The demand for a platform to store, represent and reuse the knowledge about the product and process design from various sources deemed the concepts and methodologies used in Knowledge Based Systems (KBS) a suitable fit to augment design automation efforts [50].

Sacks et al. [51] approached the design automation through automating design data representation and storing using parametric product modeling and data integration technologies. They developed a parametric template for defining data units with geometric, topological, and production processing information in a way that their combinations can generate various object classes which then instantiate a set of designs based on predefined rules. The system comprises of a set of knowledge modules with rules for structural design, floor plan design using different slab/column spacing combinations and generating work assembly for each structural element.

KBS development to attack design automation problems have been pursued by a number of other researchers. One of the early initiatives to use object-oriented product modeling for knowledge representation [50], introduced a Design Analysis Response Tool (DART) considering cost, functionality and manufacturing aspects of the design using a KBE methodology for the automotive industry. Other efforts applied a KBS to automate design of aircraft wiring harnesses [93], developed a KBE tool to propose appropriate

design changes throughout the finite element analysis and based on the results of the analysis [52], developed a KBS for automation of assembly design and cost estimation using an object-oriented knowledge representation method. Yet another study [53] proposed a knowledge-based framework linked to CAD product design and structural analysis software to support automation of the design of ascent assemblies and boom boxes for ship cranes.

6.5 Automated Detailed Design: Precast Concrete Connections

Connections perform a fundamental role in buildings and infrastructure construction. They transfer loads and stabilize the structure. There are a broad range of factors that affect design of connections. Hence, it is essential to consider all these factors that influence determination of the applied loads and other design aspects of connections. According to the PCI Connections Manual [88], the major connection design and performance criteria include strength to transfer the subjected forces, ductility, durability, fire resistance, tolerance, aesthetics, seismic requirements and constructability.

Connections and their quantity, type and design play an important role in determining cost of the construction projects. Observation of the QTO and CE practiced process in several precast concrete companies showed that currently the main guidelines for estimators in forecasting the quantity and type of connections in each project is their judgment of design situations and rules of thumb that they have developed based on their experience. As mentioned earlier variety of factors affect the design of connections and the purpose of developed rules of thumb by estimators is to simplify the process which is to a great degree manual and time-consuming. This simplification of connection design process

many times leads in not contributing some of the design features and conditions that affect the design of connections and results in less than accurate estimations.

A prototypical solution is proposed here that integrates the KBS concept with detailed QTO and CE methods as well as semantic enrichment of design models to forecast the type and quantity of connections required for precast concrete elements for various design situations. Figure 6.1 provides a comparison between the current and proposed QTO and CE practice. Similar to examples provided in Chapter 4 and 5 the inferred knowledge is added to design models to make it accessible for users.

The knowledge required for the predictive design of precast concrete connections has been acquired from precast concrete industry guidelines for connection design [86, 87, 88, 89, 90], extended interviews with several industry experts including some of the members of the Precast Concrete Institute (PCI) connections committee as well as studying company developed connection design and detailing standards and historical data from example projects provided by collaborating companies.

Current process	Proposed process
<ul style="list-style-type: none">• 2d drawing based• manual• error-prone• time-intensive• lack of transparency• over simplification	<ul style="list-style-type: none">• 3D parametric model• semi-automatic• enhanced consistency and accuracy• time-saving• enhanced transparency• Increased level of detail

Figure 6.1: Current versus proposed QTO and CE process

It is important to note that many structural engineers develop creative connection solutions especially when standard design solutions don't address the limitations and demands of extraordinary design situations. Capturing all the possible connection design configurations is impossible. The proposed connection solutions represent the standard connection quantity and types used in majority of projects. As explained in Chapter 2, the objective of the proposed solution is not to replace domain experts. The objective is to assist them by reducing the time and cost required for manual product detailed design, QTO and CE processes by automating repetitive, non-creative design tasks and by supporting multidisciplinary integration of knowledge to increase the efficiency and accuracy of results. Hence, the proposed connection design schemes for different situations reflect the typical solutions mostly practiced in the industry.

In this chapter, through a series of examples, impact of building design, relative positioning of building elements, location of element interfaces, aesthetics, constructability and erection considerations on the number and type of connections are discussed. Detailed connection assembly configuration design to meet the required strength involves complete analysis of the design loads which is out of the scope of this effort. Such a detailed structural analysis and design is performed in later stages of projects before product fabrication. The goal here is to provide enough information about the type and quantity of connections among different design elements to enable automatic QTO and CE of connections. Construction companies develop and maintain a database of unit cost for standard types of connections. Thus, when estimators are provided with the number and type of connections and their location i.e. between which design elements they occur, they

can pull the unit cost for the standard configurations used for that connection type and ultimately can calculate the total cost of connections.

6.5.1 Introduction to Precast Concrete Connections

Various criteria can be used to categorize the connections used among the precast concrete elements. In a high level and in terms of the structural role, connections can be divided to (i) gravity or bearing connections that transfer vertical loads and require a bearing surface; (ii) tie-back connections that can play various roles and provide tension, compression, torsion, shear and moment resistance; and (iii) tie-back plus gravity connections that are a combination of the first two connection types. This high level classification is used by many industry practitioners to describe the connections and is used in the provided examples in this chapter.

Various load transferring devices are used in design of connections. Some of the major categories of these devices include [86] concrete anchors or studs, deformed bar anchors and post-installed anchors like grouted anchors, rebar couplers and splice devices, bolts and threaded connectors and threaded rods. Examples of these devices are provided later in this chapter. These devices use welding or mechanical load transfer mechanisms.

6.6 Prototypical Implementation of the Proposed Solution

First all the precast concrete building elements and their interfaces with each other where they require connections were identified. Then for all element interfaces that need connections a human-readable guideline to determine number and type of connections was developed using the knowledge acquisition sources explained in section 6.5. This guideline identifies the design variables that affect the decision-making about connection designs.

Table 6.1 shows an example developed guideline. Then, several computer interpretable libraries of rule sets were developed and tested using the SEEBIM solution.

A categorization of operators and attributes in SEEBIM and their functions was discussed in detail in Chapter 4. Figure 6.2 shows all the object attributes and operators of different categories used for precast concrete connection design. The operators in boxes color-coded with purple headings are used in the IF clause of the rules to analyze the object

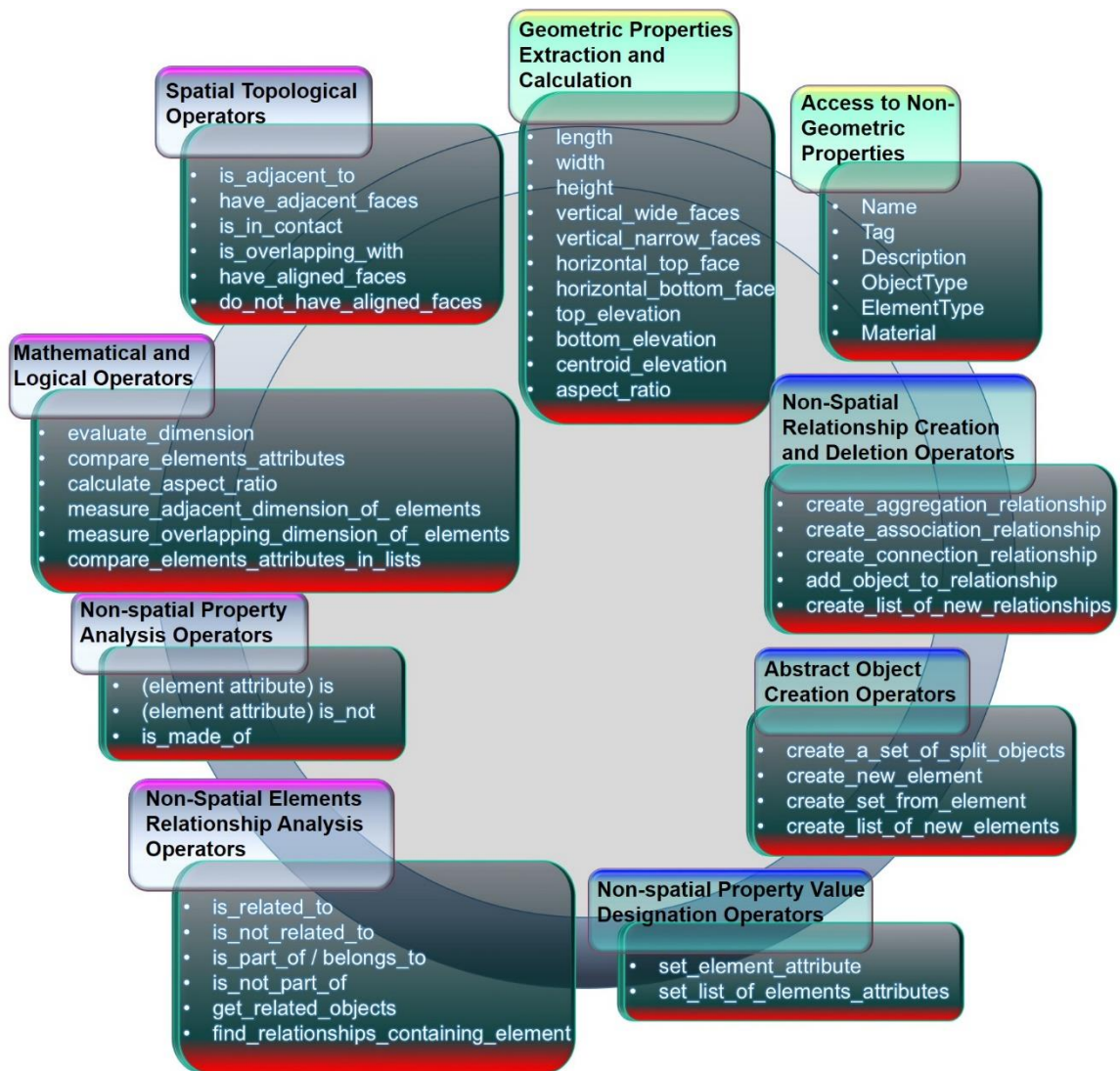


Figure 6.2: List of attributes and operators used in the predictive design of precast concrete element connections

attributes and spatial and non-spatial relationships of objects. Those in boxes color-coded with blue headings are used in the THEN clause of the rules to perform a task and add the inferred semantics to BIM-based design models. As shown in the figure, spatial topological relationships of adjacency, alignment and overlap as well as several mathematical operators to analyze element dimensions are used here to evaluate the designed objects.

Testing of the developed rule sets was performed on various precast concrete design models representing different frequently used design situations to infer quantity and type of connections in the following categories:

- Column to column connections
- Beam to column connections
- Spandrel to column connections
- Double-tee (DT) slab to double-tee connections
- DT slab to beam connections
- DT connections to shear walls parallel and perpendicular to DT direction
- Shear wall to shear wall connections

These categories contain major types of connections among structural precast concrete elements.

6.6.1 Column to Column Connections

Chapter 4 explained implementation of an algorithm developed for determining the best place for segmenting the precast concrete columns that their height exceeds the maximum feasible height determined by users. For those columns that were required to be segmented, new column pieces were created using the *create_a_set_of_split_objects* operator. This operator also creates a relationship with the *domain_type* value of *split_pieces* between the

created split column pieces and their parent column object so that the user examine which column pieces belong to which parent column.

The split column pieces need two connections in between, usually designed to be hidden: One acts as a tension, compression, shear and moment resisting connection and the other is a bearing pad that acts as a gravity connection. The first connection is a mechanical connection that usually is designed using one of the two major methods of (1) grouted splice sleeve coupler, or (2) bolted connections using anchor rods and a plate.

Figure 6.3 depicts the rule developed for creating column to column connections. A uniform color code for representing different components of the rules is used in the pseudo codes and the legend to the color code is provided in Chapter 4, Figure 4.7.

When split column pieces were created their *ObjectType* was set to *split_pieces* so that they are distinguishable from the parent columns. Hence, the rule selects the objects based on their *ObjectType*. Next to verify that both selected columns are split pieces of the same parent column, the rule checks to see if they are related by a relationship of *domain_type split_pieces*.

When the conditions are met, the rule is triggered and in the THEN clause creates the first connection using the *IfcDiscreteAccessory* entity with *Name*, *Description* and *ObjectType* values that reflect the type of connection. Then *create_connection_relationship* operator creates a connection relationship between the created connection and the column pieces. *IfcRelConnectsWithRealizingElements* is used to create the relationship which requires the realization of the relationships by its *RealizingElements* attribute. *RealizingElements* can be a set of objects that are used to represent the connections created between two elements. In order to add the created

connection to the *RealizingElements* attribute of this relationship first a set from the created connections is made. The two column pieces are added to *RelatingElement* and *RelatedElement* attribute of the relationship. Next the second connection is created and added to the relationship using the *add_object_to_relationship* operator.

6.6.2 Beam to Column Connections

Inverted-tee beam (ITB), L beam and rectangular beam are the three main types of precast concrete beams. Beams generally transfer the floor loads to the interesting columns through the beam to column connections. In general beams either intersect with one side of columns or transfer the loads to the top surface of the columns. Table 6.1 shows the guidelines developed for predicting the type and number of connections in each design situation.

First a classification rule distinguishes the spandrel beams from non-spandrel beams. Then beam to column connection creation rule, selects a non-spandrel beam and a column. Next it checks to see which faces of the two objects are adjacent: If one of the *vertical_narrow_faces* of the beam is adjacent with one of the vertical faces of the column, it means that the beam is intersecting with the side of the column. If *horizontal_bottom_face* of the beam is adjacent with the *horizontal_top_face* of the column, then beam is intersecting with the top of the column. In each situation, the appropriate number and types of connections are created in the THEN clause. The structure and function of the THEN clause is similar to the column to column connection rule.

6.6.3 Spandrel to Column Connections

Spandrels might be non-load bearing (NLB) or load-bearing (LB). NLB spandrels provide support in front of seismic forces, wind and other environment factors and carry their self-

Beam (Inverted Tee, L and Rectangular) to Column Connections	
Rules	<p>(1) All 3 types of beams (Inverted tee, L and Rectangular) to the side of column:</p> <ul style="list-style-type: none"> • Requires a corbel (aka haunch, bracket) • 3 connections in total: 1 connection in the top of beam stem (web), 1 connection in bottom of the beam stem (flange) plus a bearing pad (shim) as the gravity connection <p>(2) One beam (any one of the above 3 types) to the top of column:</p> <ul style="list-style-type: none"> • no corbel required • 2 connections in total: 1 connection from top of the beam to the column plus a bearing pad <p>(3) Two beams (and one of the 3 types) to the top of column:</p> <ul style="list-style-type: none"> • no corbel required • 2 or 3 connections in total depending on the company connection counting convention: Can be counted either as 1 connection for each beam where each connection includes 1 or 2 anchor rods and 1 plate) or total 1 connection for both beams with 2 or 4 anchor rods and 1 plate.
Types	<p>Structural Role: The connection in the bottom of the beam stem (which actually happens between the beam and the corbel) acts as a gravity + tie-back connection for roll-restraint and the bearing pad is considered a gravity connection. The connection in the top of the beam stem is a tie-back connection for roll-restraint, continuity, and erection stability.</p> <p>Fabrication Type: (1) beam to side of the column: Usually welded connections are used with embedded plates in beam and column and a field welded plate between them for the top connection and a key shear plate in corbel and bearing plate in beam for the bottom connection.</p> <p>(2) beam to top of the column: mechanical bolted connections using 1-4 anchor rods and a plate/s</p>
Notes	<ul style="list-style-type: none"> • For the beam to side of the column, the top of the stem connection likely consists of embed plates in both the column and beam with a field welded plate between them • For the beam to top of the column, the connection might have 1, 2 or 4 anchor rods on 1, 2 or 4 plates. Most often, regardless of the number of plates and rods, companies count them as 1 connection.

Table 6.1: The guideline developed for predictive design of beam to column connections

weight load. Load-bearing spandrels replace the beams and receive the floor loads and transfer them to columns.

There are various design factors in addition to their structural role that affect design of the spandrel to column connections. In most design situations three connections used between each spandrel and interfacing column: One tie-back connection close to the top edge of the spandrel, one tie-back connection close to the bottom edge of the spandrel and one gravity connection in the bottom. However, the contributing design factors affect many aspects of connection designs leading to variety of connection types used. They also determine necessity of additional design features to accommodate the connections.

6.6.3.1 Impact of Spandrel Design Conditions on Predictive Detailed Design

Structural role of spandrels as well as their positioning relative to intersecting columns impact:

- Number, type, assembly detail, capacity and location of spandrel-column connections
- Spandrel design features like daps, notches, and added corbels or ledges
- Column design and its features like added corbels
- Erection sequence and necessary provisions

Therefore, it is essential that for accurate prediction of spandrel-column connections first a set of rules to be designed to determine the following:

- Whether the spandrel is load-bearing or non-load bearing
- Whether the spandrel is inboard or outboard
- Whether the spandrel is passing through the interfacing column creating a pocket (recess) in the column

- Whether the spandrel is connecting to columns at the building corners

The following section provides examples to illustrate how structural role and positioning of spandrels impact each of the above-mentioned aspects:

1. Structural Role (LB versus NLB spandrels): When spandrels receive the floor loads from double-tee (DT), hollow-core or flat slabs and transfer the floor loads to columns eliminating the need for beams, they are considered LB spandrels. Otherwise they are NLB.

- The structural role of spandrels affect the required capacity of the connections which in turn impacts specifications of connection designs. In most design cases connections with capacity of 23-33 kips are used for LB and with capacity of 9-18 kips for NLB spandrels.
- LB spandrels transfer floor loads from double-tee and other slab types to columns. Therefore, LB spandrels intersecting slabs that transfer their load to the spandrels need a bearing surface on them that is provided either by adding a ledge in the bottom edge of the LB spandrels or a set of corbels at the intersection of double-tee slab stems (Figure 6.3 (a)).
- The structural role of spandrels along with their positioning relative to exterior face of their interfacing columns (i.e. inboard versus outboard, explained in point 2) affect the access to spandrel and column surfaces for connection inserts which in turn impacts the choice of connection assemblies. Examples of design situations in which spandrels' structural role affects the design of connections include:

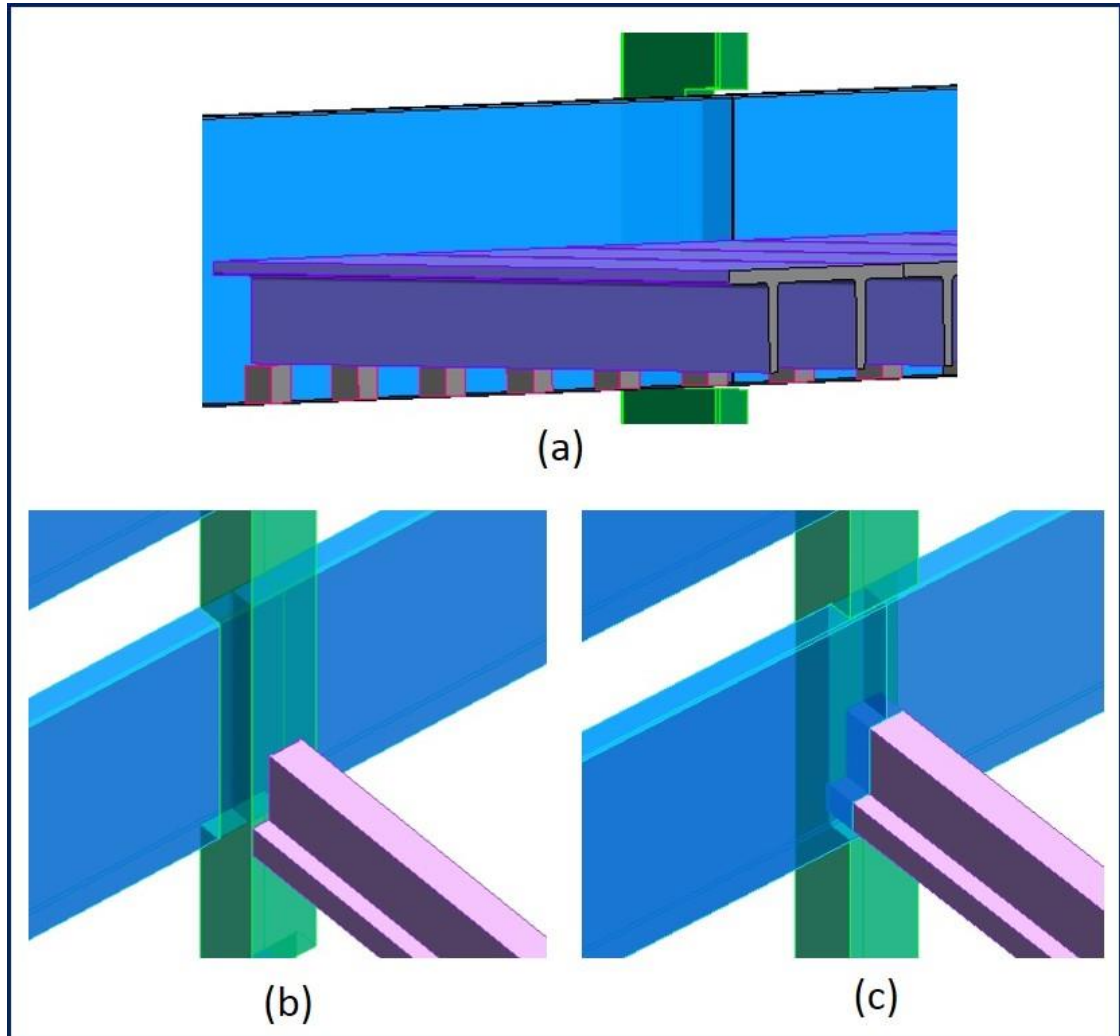


Figure 6.3: Load-bearing and non-load bearing spandrels: (a) LB spandrel with added corbels to support transfer of loads; (b) NLB outboard spandrel; (c) NLB inboard spandrel dapped to allow the intersecting beam's access to the column surface

- If the spandrel is NLB and outboard interfacing a pocketed column, the beam that intersects the column will obstruct the access required for mechanical connections with pocketed sleeves (Figure 6.3 (b)). So usually welded connections with slotted inserts are used for the bottom spandrel-column (SP-C) connections.

- If the spandrel is NLB and inboard interfacing a pocketed column, the beam that intersects the column clashes with bottom of the spandrel. Hence, these spandrels are dapped to allow the beam to pass through (Figure 6.3 (c)). Mechanical connections with pocketed sleeves can then be used for both top and bottom SP-C connections. The bottom SP-C connection will be above the bottom dap and therefore distanced further from the bottom edge of the spandrel than the bottom connections in non-dapped spandrels.
2. Inboard versus outboard: When the exterior face of spandrels is aligned with the exterior face of the intersecting columns, they are considered outboard and when the interior face of spandrels is aligned with the interior face of the intersecting columns, they are considered inboard.
- When mechanical sleeved connections are used, the pocketed sleeve and grouted surface to fill the pocket should be on the interior face hidden from outside. Therefore, if the spandrel is inboard, the sleeve and the grouted surface are placed on the spandrel and if it is outboard they will be on the column. This means that the whole connection assembly is rotated 180° depending on the spandrel's position which also affects some of the other connection assembly details like the length of the threaded or coiled rod used.
 - For LB spandrels, the distance of approximate column centroid from the axis along which DT slabs transfer their load to spandrels for outboard spandrels is much smaller than the inboard spandrels. This means that the vertical load eccentricity is smaller in outboard spandrels than inboard spandrels (Figure

6.4). This results in more tendency for the column interfacing inboard spandrels to lean out of plumb during erection, resulting in the need for special bracing and alignment during erection. Therefore, outboard spandrels are generally easier and more economical to erect.

- Connection tension and/or compression loads are identical but reversed between the two spandrel positions: For outboard spandrels, the bottom connection is in tension and for inboard spandrels the top connection is in tension. The tension condition generally governs design of the top and bottom

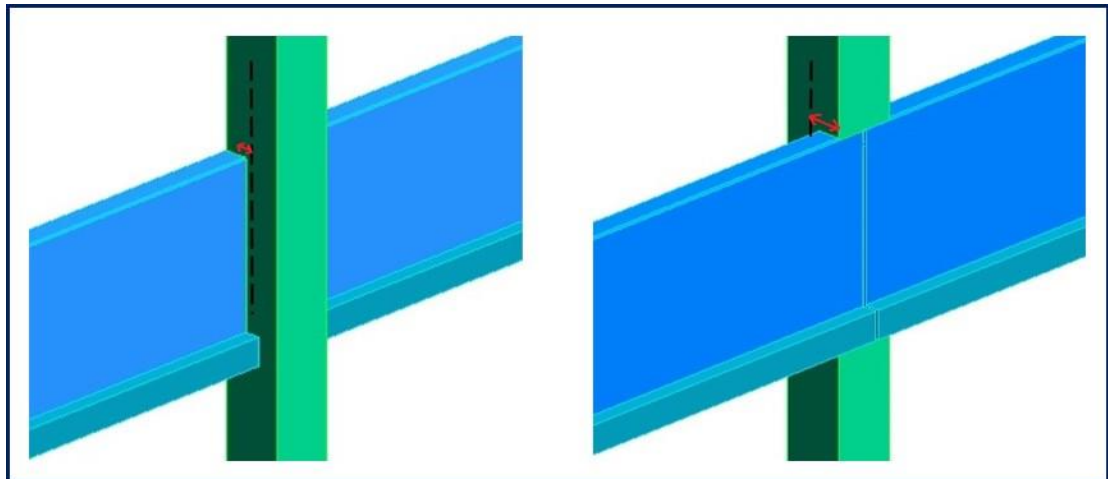


Figure 6.4: Approximate load eccentricity in outboard and inboard LB spandrels

connections. The compression connection can be simplified to a plain bearing condition if desired for economic reasons. Moreover, during the erection the tension connection should be placed first and is essential for stabilizing the spandrel.

3. Spandrels connecting to columns at the building corners versus those connecting to columns on the edge of building:

- When spandrels are at the building corner, the two spandrels intersecting the column from two sides meet at angle of 90° . This results in limitations in location of top and bottom spandrel-column connections, might impose some changes to connection assembly design and might require an added connection between the two spandrels at the corner eliminating connections between one of the corner spandrels and the intersecting column or in addition to those connections.

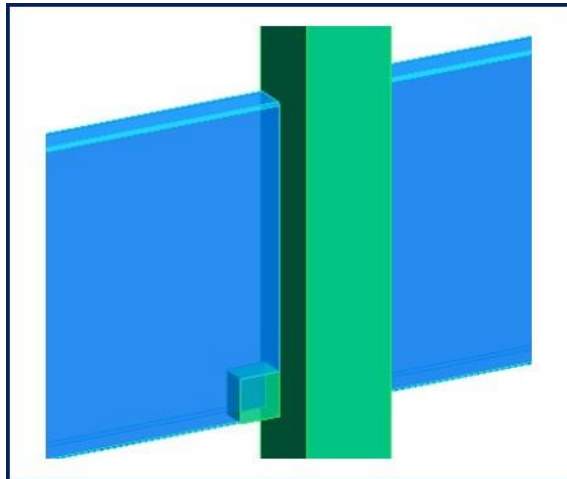


Figure 6.5: Spandrel intersecting a non-pocketed column with a notch to hide the HSS tube bracket

4. Pocketed versus non-pocketed column: When spandrels pass through the column, the column is pocketed and when they end where the column starts the column is non-pocketed.
 - For non-pocketed columns, the interfacing spandrels need some type of support with a bearing surface to accommodate the gravity connection. Thus, either a corbel is added to the column or when depending on the design situation and for aesthetic reasons the support needs to be hidden, a Hollow Structural Section (HSS) tube steel bracket embedded in the column or

welded to the column and filled with concrete or grout is used. In these situations often the bottom of the spandrel is notched to embrace the HSS tube and hide it (Figure 6.5).

- When columns are pocketed their cross-section area is reduced to a large extent. This sometimes requires designing columns with larger cross-section when they are pocketed compared to non-pocketed columns under similar design loads.
- Fabrication and erection of pocketed columns interfacing spandrels (inboard or outboard spandrels) is often more economical than non-pocketed columns despite the likelihood of having to use larger cross-section columns when they are pocketed. This is due to the forming complexity and extra cost of the added corbels on non-pocketed columns. Additionally, the pocketed columns are very tolerance insensitive, and thus these columns are easier to maintain consistent alignments during their erection.

The above considerations in design, fabrication and erection aspects in turn impact the cost of fabrication, shipping and erection of precast concrete members and ultimately the total cost of projects. Thus, it is important for all the project stakeholders to be able to quickly and reliably identify spandrels' positioning and structural role. This information is important for the constructors to more easily and accurately calculate projects' cost and be able to provide detailed design and develop product planning. It is also important for the designers to learn about the cost implications of their design choices and be able to make educated design decisions.

6.6.3.2 Rule Set Development for Predictive Detailed Design of Spandrels

The first step in developing the rules to predict the number and type of connections between spandrels and columns is to identify the previously discussed four design conditions that affect the design of connections. The methods used to identify these design conditions analyzes spatial topological relationships of the objects involved in the spandrel-column interface. The only design condition from the top four that can be identified by only analyzing direct spandrel-column relationship, is identifying whether the column is pocketed or non-pocketed: If the spandrel and column bounding boxes are overlapping, the column is pocketed. If they are adjacent, the column is non-pocketed.

Identification of other three design situations requires analyzing not only the direct relationship of spandrels and columns but also their relationship with other neighboring objects including:

- Spandrel and slab relationships (SP-SL): adjacent, overlapping or aligned in both sides or only in one side
- Column and slab relationships (C-SL): adjacent, overlapping or aligned in both sides or only in one side
- Column and beam (B-C): adjacent
- Spandrel and beam for NLB spandrels (SP-B): adjacent

These relationships change depending on the structural role of the spandrel and its relative positioning. Hence, they can be used to help identify these design conditions. Figure 6.6 and 6.7 illustrate the broad range of possible design situations for spandrels connecting to corner columns (referred to as corner spandrels) and those connected to columns. As shown in the figures some of the described object relationships change from one design situation to another.

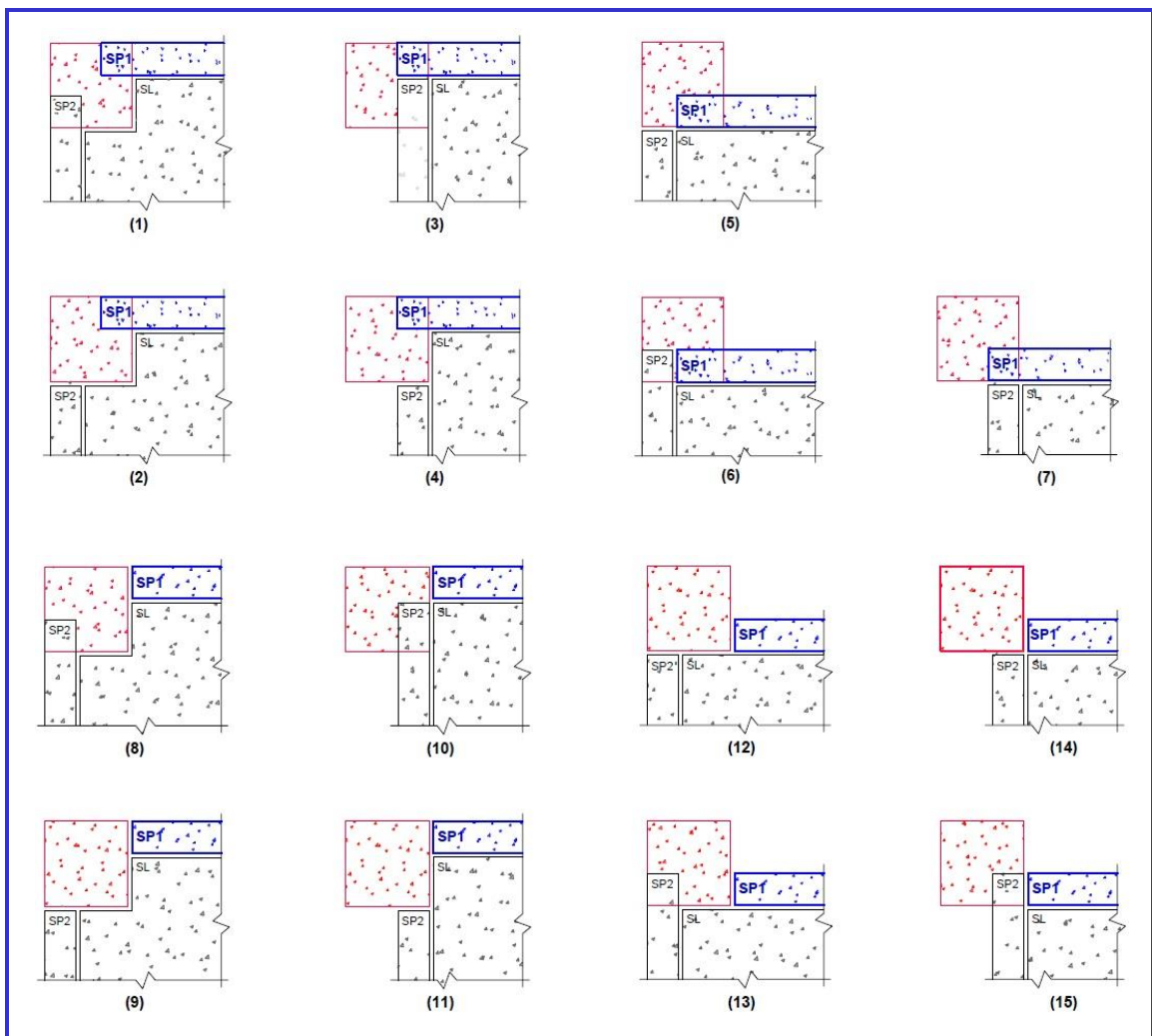


Figure 6.6: Various possible design situations for spandrels at the building corner

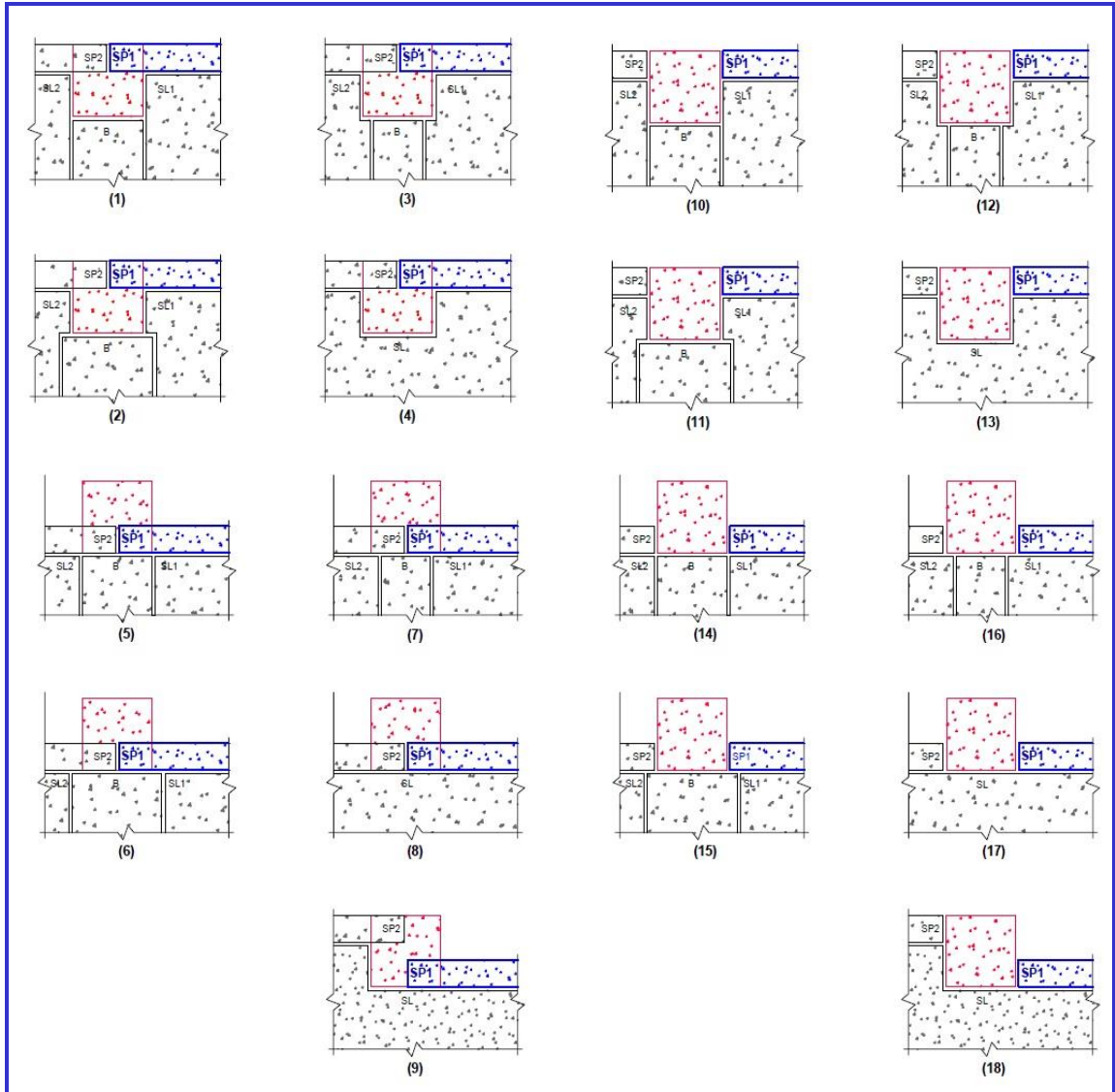


Figure 6.7: Various possible design situations for spandrels on the building edge

Design and testing of the rule sets for various design situations proved that affirmative identification of spandrel design situations in most cases require assessment of several object relationships.

As mentioned earlier in each rule a maximum of two objects can be selected. Yet complex design situations like this involve several objects and require getting access and operating on other objects that are related to the main selected objects. For this purpose first a set of basic rules were developed to identify various spatial relationships that the design objects involved in spandrel-column connections can have. The inferred result of each of these

Relationships among Building Elements		Spandrel & Column Connection Design Situations - At Building Corners														Rules	
		outboard spandrel & pocketed column				inboard spandrel & pocketed			outboard spandrel & non-pocketed column				inboard spandrel & non-pocketed column				
		1	2	3	4	5	6	7	8	9	10	11	12	13	14		15
(a)	overlapping column and spandrel	✓	✓	✓	✓	✓	✓	✓									Rule# 11-14
(b)	adjacent column and spandrel								✓	✓	✓	✓	✓	✓	✓	✓	Rule# 15-18
(c)	overlapping column and slab	✓	✓						✓	✓							Rule# 1
(d)	adjacent column and slab			✓	✓	✓	✓	✓			✓	✓	✓	✓	✓	✓	Rule# 2
(e)	aligned column and slab in both sides							✓							✓	✓	Rule# 3
(f)	aligned column and slab only in one side			✓	✓	✓	✓				✓	✓	✓	✓			Rule# 4
(g)	aligned spandrel and slab in both sides	-	-			✓	✓				✓	✓			✓	✓	Rule# 5
(h)	aligned spandrel and slab only in one side	-	-	✓	✓			✓	✓	✓			✓	✓			Rule# 6
(i)	adjacent spandrel and slab	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	Rule# 7
verify that the object related to the column is the same one adjacent or overlapping with the spandrel		slab in c = i	slab in c = i	slab in d or f = i	slab in d or f = i	slab in d or f = i	slab in d or f = i	slab in d or e = i	slab in c = i	slab in c = i	slab in d or f = i	slab in d or f = i	slab in d or f = i	slab in d or f = i	slab in d or e = i	slab in d or e = i	
Rules		Rule#11		Rule#12		Rule#13		Rule#14	Rule#15		Rule#16		Rule#17		Rule#18		

Table 6.2: Various object relationships analyzed in developing the spandrel identification and connection design rule sets for corner spandrels

Relationships among Building Elements		Spandrel & Column Connection Design Situations - Alongside Building Edges																Rules	
		outboard spandrel & pocketed column				inboard spandrel & pocketed column				outboard spandrel & non-pocketed column				inboard spandrel & non-pocketed column					
		NLB		LB		NLB		LB		NLB		LB		NLB		LB			
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16		17
(a)	overlapping column and spandrel	✓	✓	✓	✓	✓	✓	✓	✓	✓									Rule#11, 19-23
(b)	adjacent column and spandrel										✓	✓	✓	✓	✓	✓	✓	✓	Rule#15-18, 24-25
(c)	overlapping column and slab			✓	✓					✓			✓	✓					Rule#1
(d)	adjacent column and slab	✓	✓			-		✓	✓		✓	✓			-		✓	✓	Rule#2
(e)	aligned column and slab in both sides					✓									✓				Rule#3
(f)	aligned column and slab only in one side	✓	✓				✓	✓	✓		✓	✓				✓	✓	✓	Rule#4
(g)	aligned spandrel and slab in both sides			-				-			✓	✓			✓				Rule#5
(h)	aligned spandrel and slab only in one side	✓	✓	-	✓	✓	✓	-	✓	-			✓	✓		✓	✓	✓	Rule#6
(i)	adjacent spandrel and slab	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	Rule#7
(j)	overlapping spandrel and slab									✓									Rule#8
(k)	adjacent spandrel and beam					✓	✓	✓							-	✓			Rule#9
(l)	adjacent column and beam	✓	✓	✓		✓	✓	✓			✓	✓	✓		✓	✓	✓		Rule#10
verify that the object related to the column is the same one adjacent or overlapping with the spandrel		slab in d or f = i	slab in d or f = i	slab in c = (g or h)	slab in c = i	slab in e = i	slab in f = i or f = i	slab in d or f = i	slab in c or f = j	slab in d or f = i	slab in d or f = i	slab in c = i	slab in c = i	slab in e = i	slab in f = i	slab in d or f = i	slab in d or f = i	slab in c = j	
Rules		Rule#19		Rule#11		Rule #20	Rule#21		Rule #22	Rule #23	Rule#16		Rule#15		Rule #18	Rule #24	Rule#17		Rule #25

Table 6.3: Various object relationships analyzed in developing the spandrel identification and connection design rule sets for corner spandrels

basic rules are then used in final set of rules that provide an affirmative identification of spandrel design condition and create the necessary connections according to the identified spandrel design scenario.

Table 6.2 and 6.3 illustrates the object relationships used for identification of the type of spandrel in each of the 15 design scenarios of corner spandrels and 18 design scenarios of the spandrels on the building edge. As shown in the tables 12 different object

relationships marked from (a) to (l) are analyzed for slab connection design rules. For identifying the relationships of spandrels and columns with neighboring objects, relationships (c) to (l), ten basic rules, numbered as Rule#1 till Rule#10, were written. The numbers in the third row of the Table 6.2 and fourth row of the Table 6.3 denote the design situations depicted in Figure 6.6 and 6.7. The blue colored cells represent the relationships that apply in each design situation. The dark blue colored cells signify the minimum number of relationships required to affirmatively distinguish each design situation from the others. The light blue colored cells designate the relationships that although apply to those design situations, are not necessary to be used as a condition in the rules for positive spandrel type identification and connection design.

The attributes of inboard and outboard, pocketed and non-pocketed and corner and non-corner are identified for each spandrel-column interface and assigned to the connection relationship created between them. The reason is that these attributes can be different in one end of the spandrel compared to the other end of the same spandrel. This means that the spandrel for example can be inboard in one end and outboard in the other end. Hence, these are in fact attributes of spandrel-column connections. However, a spandrel can either be load-bearing or non-load bearing and the structural of a spandrel is the object attribute and as such, is assigned to spandrel object entities.

Figure 6.8 and 6.9 depict two of these ten rules, namely Rule#4 and Rule#5, developed to identify relationships of spandrels and columns with neighboring their objects. As Figure 6.8 depicts, when *vertical_narrow_faces* of the spandrel is aligned with either *vertical_narrow_faces* or *vertical_wide_faces* of the slab they are considered aligned in both sides. The reason that condition of the alignment of *vertical_wide_faces* of the


```

IF
<object1_ObjectType > is 'spandrel' AND
<object2_ElementType > is 'lfcSlab' AND
<object1> is_adjacent_to <object2>, 'tolerance', '0.1' AND
(have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_narrow_faces>, 'tolerance', '0.1' OR
have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_wide_faces>, 'tolerance', '0.1') AND
<object2> is_not_related_to <object1> relationship of 'domain_type'
  'aligned_spandrel_slab_both_sides'

THEN
create_classification_relationship 'lfcRelAssociatesClassification', between
  <object1>, <object2>, 'Name', 'Description', 'domain_type'
  'aligned_spandrel_slab_both_sides'

```

Figure 6.8: The rule designed to identify spandrel and slabs that are aligned in both sides

```

IF
<object1_ElementType> is 'lfcColumn' AND
<object2_ElementType > is 'lfcSlab' AND
<object1> is_adjacent_to <object2>, 'tolerance', '0.1' AND
(((have_aligned_faces <object1_vertical_wide_faces> are aligned with
  <object2_vertical_narrow_faces>, 'tolerance', '0.1' OR
have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_narrow_faces>, 'tolerance', '0.1') AND
do_not_have_aligned_faces <object1_vertical_wide_faces> are aligned with
  <object2_vertical_wide_faces>, 'tolerance', '0.1' AND
do_not_have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_wide_faces>, 'tolerance', '0.1') OR
((have_aligned_faces <object1_vertical_wide_faces> are aligned with
  <object2_vertical_wide_faces>, 'tolerance', '0.1' OR
have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_wide_faces>, 'tolerance', '0.1') AND
do_not_have_aligned_faces <object1_vertical_wide_faces> are aligned with
  <object2_vertical_narrow_faces>, 'tolerance', '0.1' AND
do_not_have_aligned_faces <object1_vertical_narrow_faces> are aligned with
  <object2_vertical_narrow_faces>, 'tolerance', '0.1')) AND
<object2> is_not_related_to <object1> relationship of 'domain_type',
  'aligned_column_slab_only_in_one_side'

THEN
create_classification_relationship 'lfcRelAssociatesClassification', between
  <object1>, <object2>, 'Name', 'Description', 'domain_type',
  'aligned_column_slab_only_in_one_side'

```

Figure 6.9: The rule designed to identify columns and slabs that are aligned only in one sides

spandrel with one of the vertical faces of the slab is not used in the rule is that always this condition is true. Figure 6.6 and 6.7 verify this point. The reason that the aligned face of the slab can be either of narrow or wide vertical faces is that the slab direction can be either parallel or perpendicular to the spandrel direction. In the first situation the wide vertical faces will be aligned with the narrow faces of the spandrel and in the second situation the narrow vertical faces will be aligned with the narrow faces of the spandrel. Finally the reason for adding the adjacency condition to the rule is that faces of two objects that belong to two different floors or are in different parts of the same floor can be aligned. So alignment does not verify that the two selected objects are neighboring objects. Adding the adjacency condition verifies that the two selected objects are also each other's neighbors which are the only objects of interest in the rules.

Figure 6.9 depicts Rule#5 that identifies columns and slabs that are aligned only in one sides. Part I of this rule verifies that when one of the *vertical_narrow_faces* of the selected slab is aligned with one of the vertical faces of the column, *vertical_wide_faces* of the slab are not aligned with any of the vertical faces of the column. Part II of the rule verifies that when one of the *vertical_wide_faces* of the selected slab is aligned with one of the vertical faces of the column, *vertical_narrow_faces* of the slab are not aligned with any of the vertical faces of the column. When Part I or Part II of the rule holds true for the selected slab and column, it means that they are aligned only in one side.

The result of the ten basic developed rules is creating a relationships between the two selected object according to the examined object relationship in the rule. These relationships are then called in Rule#11 till Rule#25 by using one of the *is_part_of* or *belongs_to* operators.

IF

<object1_ObjectType> <i>is</i> 'spandrel' AND <object2_ElementType> <i>is</i> 'lfcColumn' AND	(1) Main Objects Selection
--	----------------------------

<object1> <i>is_overlapping_with</i> <object2> AND	(2) Main Objects Relationship Examination
--	---

<object1> <i>is_part_of</i> relationship of 'domain_type' 'adjacent_spandrel_slab' AND <object2> <i>belongs_to</i> relationship of 'domain_type' 'overlapping_column_slab' AND	(3) Main Objects Relationships Examination with Other Objects
---	---

<i>get_related_objects</i> in relationship of 'domain_type' 'adjacent_spandrel_slab' AND <i>get_related_objects</i> in relationship of 'domain_type' 'overlapping_column_slab' AND <i>compare_elements_attributes</i> related objects in relationship 'adjacent_spandrel_slab' 'part21_line' '=' related objects in relationship 'overlapping_column_slab' 'part21_line' AND	(4) Identity Verification for Other Objects in Relationship with the Main Objects
--	---

<object2> <i>is_not_related_to</i> <object1> relationship of 'domain_type', 'outboard_spandrel_pocketed_column'	(5) Rule Execution Restriction to Once for Main Objects
--	---

THEN

<i>create_new_element</i> 'ElementType', 'lfcDiscreteAccessory', 'Name', 'domain_type', 'bearing_pad_SP_C', 'Description', 'gravity_connection_spandrel_to_pocketed_column', 'ObjectType', 'shim_connecting_spandrel_to_pocketed_column' AND <i>create_set_from_elements</i> from created elements above AND	(6) First Connection Object Creation and Addition to the Connection Relationship
--	--

<i>create_connection_relationship</i> 'lfcRelConnectsWithRealizingElements', <i>between</i> <object2>, <object1>, 'Name', 'ConnectionType', 'outboard_spandrel_pocketed_column', 'Description', '2_tie_backs_and_shim_connecting_outboard_SP_to_pocketed_C' AND	(7) Relationship Creation among the Main Objects and Connections
--	--

<i>create_new_element</i> 'ElementType', 'lfcDiscreteAccessory', 'Name', 'ObjectType', 'domain_type', 'top_tie_back_connection_SP_C', 'Description', 'top_tie_back_outboard_spandrel_to_pocketed_column' AND <i>add_object_to_relationship</i> add to 'RealizingElements' AND <i>create_new_element</i> 'ElementType', 'lfcDiscreteAccessory', 'Name', 'ObjectType', 'domain_type', 'bottom_tie_back_connection_SP_C', 'Description', 'bottom_tie_back_outboard_spandrel_to_pocketed_column' AND <i>add_object_to_relationship</i> add to 'RealizingElements' AND	(8) Connection Objects Creation and Addition to the Connection Relationship
--	---

Figure 6.10: The structure of the rule designed to identify and create outboard_spandrel_pocketed_column connection relationships and to create the required connections

Figure 6.10 illustrates the structure of Rule#11 which can be used both for corner and edge spandrel-column connections. The structure of this rule is representative of rules 11-25. In these rules first the relationship between the selected objects are examined. Then it is checked to see if the other required relationships between selected objects and other objects involved in the design situation holds true. In Rule#11 for example *adjacent_spandrel_slab* and *overlapping_column_slab* are required to hold true. The *is_part_of* and *belongs_to* operators check if the selected objects are at least in one such a relationship. But these might be involved in the designated relationship type with many objects in the model.

Hence, it is important to verify that the third object involved in the relationships with the main objects is in fact the same object. In this example it is verified that the slab adjacent to the spandrel and overlapping with the column is the same slab. This verification is performed through *compare_elements_attributes* operator and by examining the *part21_line* number of the related objects in those relationships. Part 21 files are IFC files. In these files each object instance has a unique line number that can be used for the verification of identity of design entities. When *part21_line* number of the related objects in those two relationships is equal, it means they are in fact the same object. The logic and structure of the THEN clause of these rules is similar to column to column connection rules explained in section 6.6.1.

6.6.4 Double-Tee, Shear Wall and Beam Connections

Figure 6.11 illustrates the design situations where the connections between two DTs, between DTs and shear walls, between DTs and beams and between two shear walls can

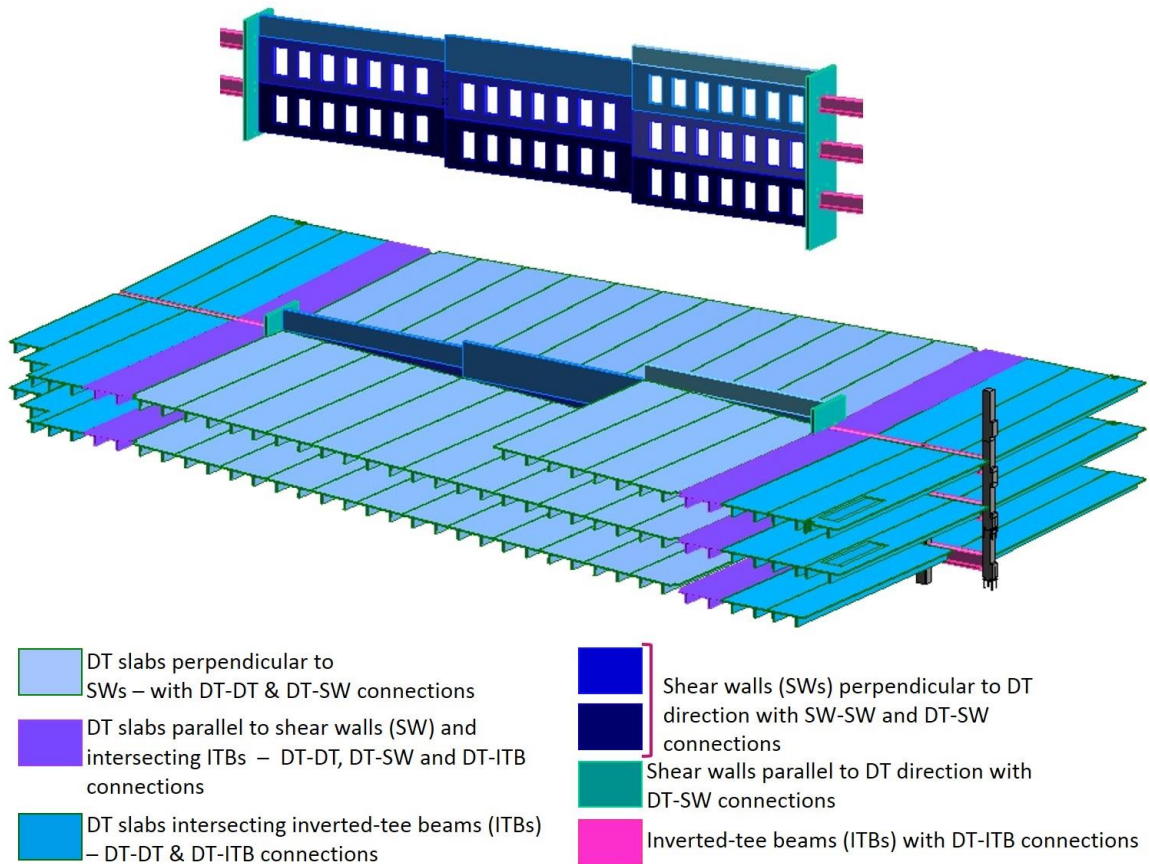


Figure 6.11: Design situations for Double-tee, shear wall and beam connections (model courtesy of The Consulting Engineering Group company)

happen. The model depicted in this figure is also used as a test model for the rules of this section. Similar to previous sections, the rules in this section examine the spatial topological relationships of objects to identify the type and number of connections used between them. Results of running the rule sets are added to IFC files to create enriched models. As illustrated in Figure 6.12, these added connections can be seen by users when the enriched IFC models are imported to Autodesk Navisworks Manage software. They

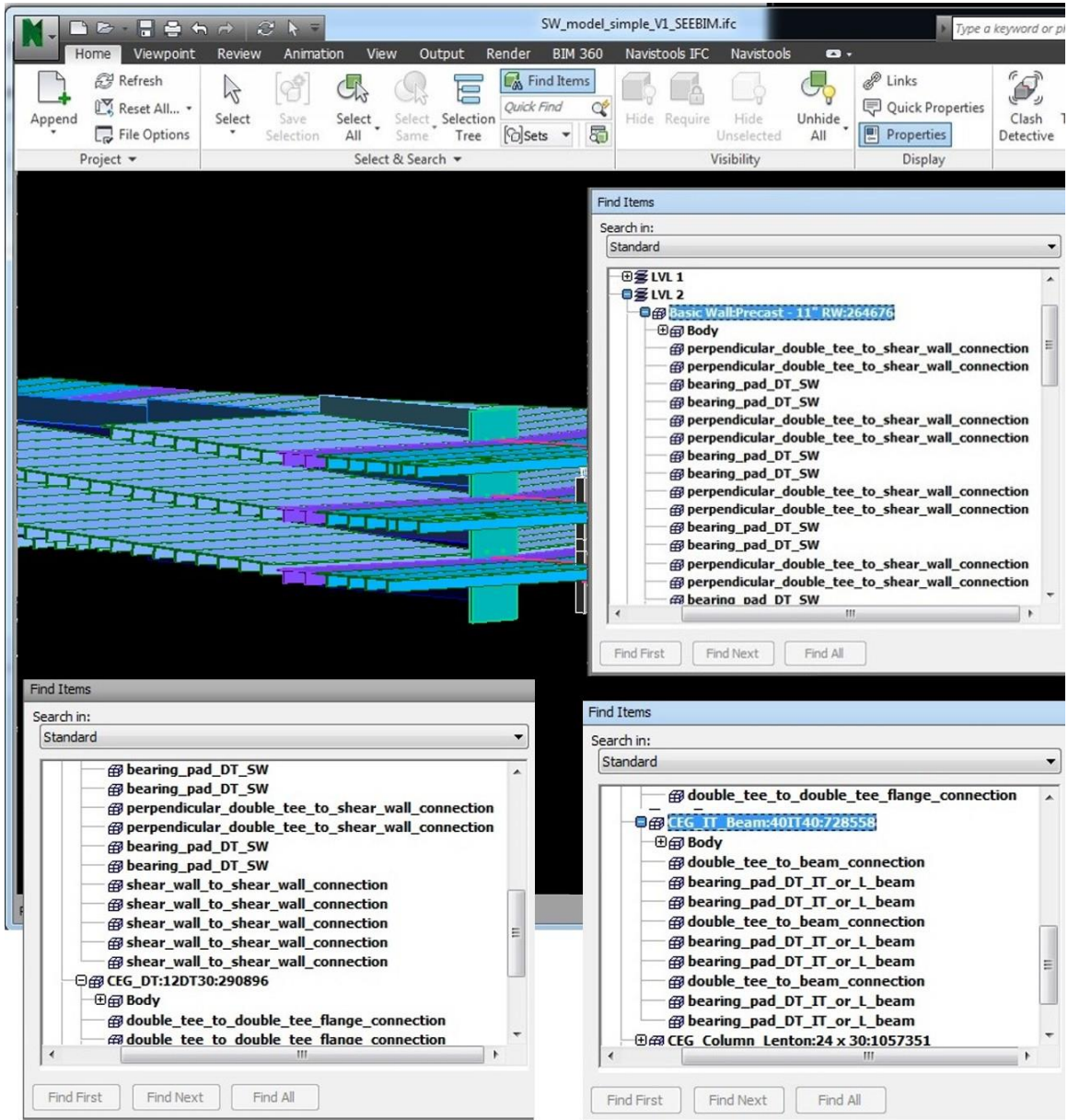


Figure 6.12: The enriched IFC model imported to Navisworks Manage that depicts the added double-tee, shear wall and beam connections

can be found in Find Items window and Selection Tree window that provide a breakdown structure of the objects in the model.

CHAPTER VII

LIMITATIONS AND GENERALIZATION OF THE PROPOSED FRAMEWORK

7.1 Research Limitations

Limitations of this research work can be discussed from the technical point of view as well as industry implementation point of view.

From the technical implementation standpoint the first limitation is using a simplified object geometry based on object bounding boxes to develop and test rule sets. Hence, features like recession, blockout or dap in objects don't impact their relationships with other objects since they don't impact the geometry of its bounding box. The experience of solving several problems using this system showed that simplifying object geometry to its bounding box sometimes have been helpful for solving problems and sometimes didn't provide complete information about an object and required developing a workaround. Moreover, this system can only be applied to objects with rectangular shapes and spatial relationships of curved or otherwise free-form objects cannot be handled within this system. This limitation was not significant in the test domain of the structural precast concrete since most objects have standard shapes. Yet, it will impose an important limitation for extending the system to areas like architectural precast concrete.

Moreover, in the current implementation, the created new objects like column and slab pieces, tendons, connections and corbels don't have a geometric representation or placement. In the framework of using design models for QTO and CE this is not an

important drawback since it doesn't impact the ability to extract the quantities of objects. However, automatic addition of geometric representation for created objects using the previously developed MVDs will support expanding the use of the enriched models directly for other downstream activities like detailed design and production planning.

The vision discussed in detail in Chapter 5 considers the KBS a platform that has access to and can use the information created by different design, analysis and project management platforms in order to extract knowledge, infer new knowledge and present it to users. Yet, due to lack of interoperability among different tools used in a project lifecycle, in the current test cases the information output of each tool was manually imported in the other tool. This of course is a long-discussed problem and many research teams and industry organizations attempt to solve it.

7.2 System Generalization

System generalization can be discussed both in terms of implementing the developed framework across the industry in the domain for which the prototypical solution was developed and in terms of expanding the proposed methodology to other domains in the AEC industry. Several strategies during the research work was used to mitigate the limitations of generalizing the developed framework.

Lack of standardization which is one of the characteristics of the AEC industry, poses a great challenge to this research: Various processes and rules are practiced in different parts of the industry. The problem is how to develop the rules so that they represent a wide array of options and approaches practiced by different industry practitioners.

The first step to handle this challenge was to consult with different companies of different sizes, in various levels of technology adoption and in diverse segments from trade companies to engineering consulting companies to general contractors. This helped to define the problems from different points of view and build a wide vision about the processes and solutions deployed in different segments. Also the results of each step of the work explained in the methodology segment was checked and verified with representatives of different companies and sectors.

Another fundamental approach used to mitigate the impact of nonstandard industry solutions, was to define a minimum industry-wide core concept for each step of the problem solving algorithms. This minimum concept included the common practice that was accepted by representatives of different companies. Differences in company practices that reflected company production limitations and preferences were represented by variables which are parameters that users can select, and tweak and adjust their values to reflect their project or company conditions and preferences.

This can be explained using the example of column segmentation: One shared core concept is max column length that is feasible to fabricate and erect or otherwise economically practical and hence preferred. Yet, this length can be different for different companies based on their production plant and available trucks and sometimes specific conditions of each project might impose setting a different maximum length for each project. So instead of setting a specific number for all users, “max feasible column length” is identified and incorporated in rules as a variable for which the users can provide their selected value.

Another core concept is segmenting columns in closest location possible to the middle of the column and that the closest location to the middle of column is defined in relationship with intersecting floors or spandrels. The preferred splicing location for instance for internal columns, is 2' above the finished floor of closest intersecting floor to the middle of the column. While this is practiced by majority of the companies, the location compared to finished floor depends of the building design and connection types used. Hence, it is in a range usually between 1.5'-2' above the finished floor and cannot be presented by one number. Hence, again this range is represented as a variable in the rule sets not as fixed number.

So basically the design rule differences are identified and represented as a set of variables so that the rules and their outcome can be easily adjusted to represent preferences of different users. Using this method throughout the rule development ensures flexibility of the system and applicability to a wide array of practices.

7.3 System Extendibility to Other Domains

There are many fundamental similarities in the supply chain and information workflow of different building systems. While the knowledge body and content of the rules are different for different domains, the proposed architecture, methodology and fundamental building blocks of the system can be reused to expand its applications to other domains in the AEC industry.

If we consider for instance other building structure systems including CIP concrete and steel structures, analogies in the preconstruction processes, the sub-functions and the type of information required exist among them. For instance, forecasting type and number

of connections among different objects or segmenting the objects into constructible modules are also required for steel structures. In the precast concrete and steel these modules are product pieces and in CIP concrete they are concrete pours which act as a type of connection. Of course, the supply chain process and different structural properties of different systems impose different rules for each system and the knowledge for building the rules need to be investigated. But to a large degree they all use the same fundamental concepts and information items.

These reusable building blocks include the concepts developed to define geometric and non-geometric attributes of each product type, the concepts developed to define various spatial and non-spatial relationships among objects, and flexible rule structures that use these shared attribute and object relationship concepts which can be mixed and matched to customize the rules and build virtually infinite number of rule sets.

CHAPTER VIII

BROADER IMPACTS OF THE RESEARCH AND CONCLUSIONS

This research effort proposed a framework for a knowledge-based system integrated with parametric object-oriented modeling platforms to support and streamline BIM-based preconstruction activities. The focus was on providing a framework for acquiring, structuring, representing and reusing the domain experts' knowledge and inferring new knowledge to be used in downstream project activities. The knowledge base includes process maps, product decomposition models and elucidation of required information items, the flow of information throughout these activities that simulates the process adopted by industry experts.

The simulated expert processes were then represented as a set of problem solving algorithms, based on which modularized libraries of rule sets were created. First category of rule sets semantically enhance design models by embedding the identified design information required for preconstruction activities. The enhanced design models are then used for modularization of the design objects into elements that can be fabricated and erected. Finally the last module is applied to the modularized and prepared design model that includes the rule sets designed to automatically predict the product features and those attributes that are missing from the design and to automatically add them to the design models. These rule sets are developed to discover and embed geometric and non-geometric

attributes of design objects, to detect spatial topological relationships among objects, and to create new logical objects and various relationships between objects.

The industry experts contributed to the project all emphasized the necessity of developing such a knowledge-based automation system and the potential benefits for the industry. They have been consulted with about outcomes of each step and their comments and modifications were reflected in the developed models and rule sets. The methodology and building blocks of the system can be reused for developing BIM-based automated preconstruction activities in other domains of the AEC industry.

- *Streamlining flow of information from BIM-based design to preconstruction activities.* Currently there is a misalignment of object representation in design models compared to the object representation in the form of constructible modules required for preconstruction and construction activities. Due to this misalignment, construction entities often have to develop models from scratch. This research work proposes a framework that evaluates the designed objects based on the defined rules and when necessary creates new objects with constructible geometry and provides their quantity information to users. As a result the need to create new models are eliminated and design models can be directly used as the base model for elaborated detailed models used in construction activities.

- *Semi-automating preconstruction activities.* Currently even when BIM is adopted for preconstruction activities, since many object features and design elements important for accurate QTO, CE and other construction activities are absent with a potential to improve accuracy, they need to be manually forecasted and accounted for. Through semantic enrichment and automated detailed design the proposed framework

automatically forecasts the missing features and elements of the design and adds them to the model.

- *Improving the cost-effectiveness of adopting BIM in preconstruction activities.*

This is the result of eliminating the need to create models from scratch for preconstruction purposes as well as semi-automating the process. The time-saving resulted from these two improvements will make the adoption of BIM for these activities economically viable. Currently due to the fact that these activities are labor-intensive and also that companies only win a fraction of the projects that they bid for, and to make the process economically practical, many of the estimations rules are simplified and some design conditions are not accounted for in estimations.

Through automating the repeated and time-consuming tasks during the preconstruction and detailed structural design stage, the proposed framework enables the industry practitioners to focus on creative aspects of these activities and optimizing the design. Moreover, it facilitates accounting for more design conditions in their estimations and provide more detailed estimations and potentially improving the accuracy of cost estimations.

- *Communication enhancement.* A KBS for BIM-integrated preconstruction activities will provide a visual medium to streamline communication of the logic and intent of project cost estimation with different entities in a project lifecycle from architects to structural engineers, plant managers and general contractors. Right now the applied estimation process and rules are not well communicated among different project parties and sometimes even inside one company. Using this system trade contractors can more efficiently communicate their estimation logic with general contractors. Even though

preconstruction experts try to consult with structural engineers on unusual design situations, as it is done through traditional time-consuming methods, collaboration between them is limited and unstructured. Many times there is a disconnection between actual structural design and estimators' assumptions.

This system provides a two way systematic communication between structural designers and estimators, where structural engineers can, to the degree possible, follow in their design the same logic used in the estimation. This way the actual cost of a project will be kept more in line with the estimated cost. And when structural limitations don't allow this to happen, they can provide feedback through the system to alter and improve the estimation rules. Hence, a continuous and virtuous feedback and improvement loop between structural designers and estimators will be created.

- *Paradigm shift in knowledge availability.* In the proposed solution, the detailed structural design, fabrication and construction knowledge was encapsulated and the inferred knowledge was provided through enriched design models. Hence, implementation of this methodology will facilitate capturing construction and disseminating this knowledge to both construction entities as well as designers and other parties involved in the AEC projects. This will shift the availability of detailed structural design and construction process information to earlier in the project lifecycle and during conceptual design and design development. Such a shift in knowledge availability will create a new paradigm where architects and structural engineers of record can in real time see results of their design decisions on constructability and cost of a project and can instantly modify and improve their design rather than waiting until late project stages when changes in the design will be more costly.

APPENDIX A

RULE SET FOR AUTOMATIC SEGMENTATION OF PRECAST CONCRETE COLUMNS

```
int x;
bool why;

for (int i = 0; i < Element_list.Count ; i++)
{
    if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcColumn") &&
        !Operators.if_is_a( Element_list[i], "ObjectType", "column_segmented") &&
        Operators.is_made_of( Rel_list, Element_list[i], "concrete") &&
        Operators.is_dimension( Element_list[i], "height", ">", 50) &&
        Operators.is_dimension( Element_list[i], "height", "<=", 100 )
        { if ( Operators.change_element_attribute( Element_list, Element_list[i],
            "ObjectType", "column_segmented") &&
            Operators.split_up( Element_list, Rel_list, Element_list[i], "2")
            { Flag=true;
            System.Console.WriteLine("Rule #1.: i = " + i );}
        }

    if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcBeam") &&
        Operators.is_made_of( Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio( Element_list[i], "height", "width") &&
        Operators.is_dimension( Element_list[i], "aspect_ratio", "<", 4.6) &&
        !Operators.if_is_a( Element_list[i], "ObjectType", "non_spandrel_beam" ))
        { if ( Operators.change_element_attribute( Element_list, Element_list[i],
            "ObjectType", "non_spandrel_beam") &&
            Operators.change_element_attribute( Element_list, Element_list[i],
            "Tag", "non_spandrel_beam" )
            { Flag=true;
            System.Console.WriteLine("Rule #2.: i = " + i );}
        }

    if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcBeam") &&
        Operators.is_made_of( Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio( Element_list[i], "height", "width") &&
        Operators.is_dimension( Element_list[i], "aspect_ratio", ">=", 4.6) &&
        !Operators.if_is_a( Element_list[i], "ObjectType", "spandrel" ))
        { if ( Operators.change_element_attribute( Element_list, Element_list[i],
            "ObjectType", "spandrel") &&
            Operators.change_element_attribute( Element_list, Element_list[i], "Tag", "spandrel" )
            { Flag=true;
            System.Console.WriteLine("Rule #3: i = " + i );}
        }

    for (int j = 0; j <= Element_list.Count; j++)
    {
        if ( j == Element_list.Count )
        { break; }
        if ( i == j ) continue;
    }
}
```

```

List<DB.IFCArray> elements = new List<DB.IFCArray>();
    DB.IFCArray element = new DB.IFCArray();
    DB.RelObj relationship = new DB.RelObj();
    DB.RelObj relationship1 = new DB.RelObj();
    DB.RelObj relationship2 = new DB.RelObj();

    if ( Operators.if_is_a( Element_list[i], "ObjectType","non_spandrel_beam") &&
        Operators.if_is_a( Element_list[j], "ObjectType","non_spandrel_beam") &&
        Operators.is_not_related_to(Rel_list, Element_list[i], Element_list[j],
            "adjacent_beams_same_floor") &&
        Operators.has_adjacent_faces_with( Element_list[i].vertical_narrow_faces(),
            Element_list[j].vertical_narrow_faces(), 3) &&
        Operators.compare_elements_attribute( Element_list[i],
            "Top_Elevation","=",Element_list[j], "Top_Elevation") &&
        Operators.is_obj_between_exist(Element_list, elements, Element_list[i],
            Element_list[j]) &&
        Operators.filter(elements, "ElementType","lfcColumn")
    {
        if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[i],
            Element_list[j], "adjacent_beams", "adjacent_beams",
            "adjacent_beams_same_floor") &&
        Operators.change_elements_attribute(Element_list, elements,
            "Tag","internal_column"))
        {Flag=true;
        System.Console.WriteLine("Rule #4: i = " + i + " j = " + j);}
    }

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcColumn") &&
        Operators.if_is_a( Element_list[j], "ObjectType","spandrel") &&
        !Operators.if_is_a( Element_list[i], "ObjectType","split") &&
        Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
        !Operators.if_is_a( Element_list[i], "Tag","segmented_like_internal_column") &&
        !Operators.if_is_a( Element_list[i], "Tag","external_column") &&
        !Operators.if_is_a( Element_list[i], "Tag","internal_column") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag","external_column"))
        {Flag=true;
        System.Console.WriteLine("Rule #5: i = " + i + " j = " + j);}
    }

    if ( Operators.if_is_a( Element_list[i], "Element_Type","lfcColumn") &&
        Operators.if_is_a( Element_list[j], "ObjectType","spandrel") &&
        !Operators.if_is_a( Element_list[i], "ObjectType","split") &&
        Operators.is_overlapping( Element_list[i], Element_list[j]) &&
        !Operators.if_is_a( Element_list[i], "Tag","segmented_like_internal_column") &&
        !Operators.if_is_a( Element_list[i], "Tag","external_column") &&
        !Operators.if_is_a( Element_list[i], "Tag","internal_column") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag","external_column"))
        {Flag=true;
        System.Console.WriteLine("Rule #6: i = " + i + " j = " + j);}
    }

```



```

}

if ( Operators.if_is_a( Element_list[i], "Tag","internal_column") &&
Operators.if_is_a( Element_list[j], "ObjectType","spandrel") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
!Operators.if_is_a( Element_list[i], "Tag","segmented_like_internal_column" ))
{ if ( Operators.change_element_attribute(Element_list, Element_list[i],
"Tag","segmented_like_internal_column"))
{Flag=true;
System.Console.WriteLine("Rule #7: i = " + i + " j = " + j);}
}

if ( Operators.if_is_a( Element_list[i], "ObjectType","non_spandrel_beam") &&
Operators.if_is_a( Element_list[j], "Tag","internal_column") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
Operators.is_not_part_of(Rel_list, ref relationship, Element_list[j],
"closest_intersecting_beam_column" ))
{
if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "closest_intersecting_beam_column",
"closest_intersecting_beam_column", "closest_intersecting_beam_column") &&
/* Operators.change_element_attribute(Element_list, Element_list[j],
"DomainType", "column_checked_for_closest") && */
Operators.change_element_attribute(Element_list, Element_list[i], "Description",
"closest_intersecting_beam_to_column_centroid"))
{Flag=true;
System.Console.WriteLine("Rule #8: i = " + i + " j = " + j);}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "non_spandrel_beam") &&
Operators.if_is_a(Element_list[j], "Tag", "internal_column") &&
Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
Operators.belongs_to(Rel_list, ref relationship1, Element_list[j],
"closest_intersecting_beam_column") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"closest_intersecting_beam_column") &&
Operators.is_closest(Element_list, relationship1, Element_list[i], Element_list[j],
"centroid_elevation"))
{
if (Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.change_element_attribute(Element_list, elements[1], "Description",
"not_closest_intersecting_beam_to_column_centroid") &&
Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "closest_intersecting_beam_column",
"closest_intersecting_beam_column", "closest_intersecting_beam_column") &&
Operators.delete_relationship(Rel_list, relationship1) &&
Operators.change_element_attribute(Element_list, Element_list[i], "Description",
"closest_intersecting_beam_to_column_centroid"))
{
Flag = true;
System.Console.WriteLine("Rule #9: i = " + i + " j = " + j);
}
}

```

```

}

if ( Operators.if_is_a( Element_list[i], "ObjectType","non_spandrel_beam") &&
Operators.if_is_a( Element_list[j], "Tag","segmented_like_internal_column") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
Operators.is_not_part_of(Rel_list, ref relationship, Element_list[j],
"closest_intersecting_beam_column" ))
{ if ( Operators.create_rel(Rel_list, element, "lfcRelAggregates", Element_list[j],
Element_list[i], "closest_intersecting_beam_column",
"closest_intersecting_beam_column", "closest_intersecting_beam_column") &&
/* Operators.change_element_attribute(Element_list, Element_list[j],
"ObjectType", "column_checked_for_closest") && */
Operators.change_element_attribute(Element_list, Element_list[i], "Description",
"closest_intersecting_beam_to_column_centroid"))
{Flag=true;
System.Console.WriteLine("Rule #10: i = " + i + " j = " + j);}
}

if ( Operators.if_is_a( Element_list[i], "ObjectType","non_spandrel_beam") &&
Operators.if_is_a( Element_list[j], "Tag","segmented_like_internal_column") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
Operators.belongs_to(Rel_list, ref relationship1, Element_list[j],
"closest_intersecting_beam_column") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"closest_intersecting_beam_column") &&
Operators.is_closest(Element_list, relationship1, Element_list[i], Element_list[j],
"centroid_elevation"))
{
if (Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.change_element_attribute(Element_list, elements[1], "Description",
"not_closest_intersecting_spandrel_to_column_centroid") &&
Operators.create_rel(Rel_list, element, "lfcRelAggregates", Element_list[j],
Element_list[i], "closest_intersecting_beam_column",
"closest_intersecting_beam_column", "closest_intersecting_beam_column") &&
Operators.delete_relationship(Rel_list, relationship1) &&
Operators.change_element_attribute(Element_list, Element_list[i], "Description",
"closest_intersecting_beam_to_column_centroid"))
{Flag=true;
System.Console.WriteLine("Rule #11: i = " + i + " j = " + j);}
}

if ( Operators.if_is_a( Element_list[i], "ObjectType","spandrel") &&
Operators.if_is_a( Element_list[j], "Tag","") &&
(Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) ||
Operators.is_overlapping( Element_list[i], Element_list[j])) &&
Operators.is_not_part_of(Rel_list, ref relationship, Element_list[j],
"closest_intersecting_spandrel_column" ))
{
if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "closest_intersecting_spandrel_column",
"closest_intersecting_spandrel_column", "closest_intersecting_spandrel_column")
&&
Operators.change_element_attribute(Element_list, Element_list[i], "Description",
"closest_intersecting_spandrel_to_column_centroid") &&

```


APPENDIX B

RULE SET FOR AUTOMATIC MODULORIZATION OF PRECAST CONCRETE SLAB

```
for (int i = 0; i < Element_list.Count ; i++)
{
    int x = 0;
    bool why = false;
    DB.RelObj relationship6 = new DB.RelObj();
    DB.IFCArray element1 = new DB.IFCArray();

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcSlab") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&

        Operators.is_dimension( Element_list[i], "height", ">=", 2) &&
        Operators.is_dimension( Element_list[i], "height", "<=", 2.85) &&
        Operators.is_dimension( Element_list[i], "width", ">", 30) &&
        !Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "ObjectType", "non_spandrel_beam") &&
        Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag", "double_tee_slab"))
        {Flag=true;
        System.Console.WriteLine("Rule #7: i = " + i );}
    }

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&

        Operators.is_dimension( Element_list[i], "height", ">=", 1.5) &&
        Operators.is_dimension( Element_list[i], "height", "<=", 5) &&
        Operators.is_dimension( Element_list[i], "width", ">=", 2.3) &&
        Operators.is_dimension( Element_list[i], "width", "<=", 3.4) &&
        !Operators.if_is_a( Element_list[i], "Tag", "inverted_tee_beam" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag", "inverted_tee_beam"))
        {Flag=true;
        System.Console.WriteLine("Rule #8: i = " + i );}
    }
}
```

```

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcSlab") &&
    Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
    Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab" ) &&
    !Operators.if_is_a( Element_list[i], "Name", "Floor:Precast Concrete Slab - 30 inch
thick" ))
{ if ( Operators.get_element_dimension(Element_list, Element_list[i], "width", ref y) &&
    Operators.change_element_attribute(Element_list, Element_list[i], "ObjectType", "slab
width" + y.ToString() + "") &&
    Operators.change_element_attribute(Element_list, Element_list[i], "Name",
"Floor:Precast Concrete Slab - 30 inch thick" ))
{Flag=true;
System.Console.WriteLine("Rule #9: i = " + i );}
}

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcSlab") &&
    Operators.is_part_of(Rel_list, ref relationship6, Element_list[i],
"columns_supporting_the_slab") &&
    !Operators.if_is_a(Element_list[i], "DomainType",
"checked_for_number_of_supporting_columns" ))
{ if ( Operators.count_objects( relationship6, "RelatedObjects", ref x) &&
    Operators.get_relating_object(Element_list, relationship6, ref element1) &&
    Operators.change_element_attribute(Element_list, element1, "Description", "slab
passes through " + (int) Math.Ceiling(x/2 -1) + " bays") &&
    Operators.change_element_attribute(Element_list, Element_list[i], "DomainType",
"checked_for_number_of_supporting_columns" ))
{Flag=true;
System.Console.WriteLine("Rule #3: i = " + i );}
}

```

```

for (int j = 0; j <= Element_list.Count; j++)
{
    if (j == Element_list.Count)
    {break;}
    if (i == j) continue;

```

```

List<DB.IFCArray> elements = new List<DB.IFCArray>();
List<DB.IFCArray> elements1 = new List<DB.IFCArray>();
List<DB.IFCArray> elements2 = new List<DB.IFCArray>();
DB.IFCArray element = new DB.IFCArray();
DB.RelObj relationship = new DB.RelObj();
DB.RelObj relationship1 = new DB.RelObj();
DB.RelObj relationship2 = new DB.RelObj();
DB.RelObj relationship3 = new DB.RelObj();
DB.RelObj relationship4 = new DB.RelObj();
DB.RelObj relationship5 = new DB.RelObj();

```

```

List<DB.RelObj> list = new List<DB.RelObj>();
List<DB.RelObj> list1 = new List<DB.RelObj>();

```

```

float y = 0;

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType","lfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType","lfcSlab") &&
Operators.is_made_of(Rel_list, Element_list[j], "concrete") &&
(Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) ||
Operators.is_overlapping( Element_list[j], Element_list[i] )) &&
Operators.is_not_part_of(Rel_list, ref relationship, Element_list[j],
"columns_supporting_the_slab" ))
{ if ( Operators.create_rel(Rel_list, "lfcRelAggregates", Element_list[j], Element_list[i],
"columns_supporting_the_slab", "columns_supporting_the_slab",
"columns_supporting_the_slab"))
{Flag=true;
System.Console.WriteLine("Rule #1: i = " + i + " j = " + j);}
}

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType","lfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType","lfcSlab") &&
Operators.is_made_of(Rel_list, Element_list[j], "concrete") &&
(Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) ||
Operators.is_overlapping( Element_list[j], Element_list[i] )) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"columns_supporting_the_slab") &&
Operators.is_part_of(Rel_list, ref relationship,Element_list[j],
"columns_supporting_the_slab"))
{ if ( Operators.add_object_to_relationship(relationship, Element_list[i],
"RelatedObjects"))
{Flag=true;
System.Console.WriteLine("Rule #2: i = " + i + " j = " + j);}
}

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
Operators.if_is_a( Element_list[j], "ElementType","lfcSlab") &&
(Operators.has_adjacent_faces_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.1) ||
(Operators.is_overlapping( Element_list[j], Element_list[i] ) &&
Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.9)))&&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i], "column_bay"))
{ if ( Operators.create_rel(Rel_list, "lfcRelAggregates", Element_list[j], Element_list[i],
"column_pair_and_beam_supporting_the_slab", "column_bay", "column_bay"))
{Flag=true;
System.Console.WriteLine("Rule #4: i = " + i + " j = " + j);}
}

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcBeam") &&
(Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) ||
Operators.is_overlapping( Element_list[j], Element_list[i] )) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i], "column_bay")
&&
Operators.is_part_of(Rel_list, ref relationship, Element_list[j], "column_bay") &&
Operators.find_relationships_containing_element(Rel_list, Element_list[j],
"column_bay", "RelatedObjects", list)
{if (Operators.add_object_to_relationships(Rel_list, list, Element_list[i],
"RelatedObjects"))
{ Flag = true;
System.Console.WriteLine("Rule #5: i = " + i + " j = " + j);}
}

```

```

if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_related_to(Rel_list, Element_list[j], Element_list[i], "column_bay") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"checked_for_column_bay_length") &&
Operators.find_relationships_containing_element(Rel_list, Element_list[i],
"column_bay", "RelatedObjects", list)
{
if (Operators.change_relationships_attribute(Rel_list, list, "Description", " Bay
Length", Element_list[i], Element_list[j]) &&
Operators.create_relationships(Rel_list, list, "lfcRelAssociatesClassification",
Element_list[j], Element_list[i], "checked_for_column_bay_length",
"checked_for_column_bay_length", "checked_for_column_bay_length"))
{Flag = true;
System.Console.WriteLine("Rule #6: i = " + i + " j = " + j);}
}

```

APPENDIX C

RULE SET FOR AUTOMATIC DESIGN OF COLUMN TO COLUMN AND COLUMN TO BEAM CONNECTIONS

```
for (int i = 0; i < Element_list.Count; i++)
{
    if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
        !Operators.if_is_a(Element_list[i], "DomainType", "segmented") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
        Operators.is_dimension(Element_list[i], "height", ">", 30) &&
        Operators.is_dimension(Element_list[i], "height", "<=", 60))
    {if (Operators.change_element_attribute(Element_list, Element_list[i], "DomainType",
        "segmented") &&
        Operators.split_up(Element_list, Rel_list, Element_list[i], "2"))
        { Flag = true;
            System.Console.WriteLine("Rule #1.1: i = " + i);
        }
    }

    if (Operators.if_is_a(Element_list[i], "ElementType", "lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio(Element_list[i], "height", "width") &&
        Operators.is_dimension(Element_list[i], "aspect_ratio", "<", 4.6) &&
        !Operators.if_is_a(Element_list[i], "ObjectType", "non_spandrel_beam"))
    { if (Operators.change_element_attribute(Element_list, Element_list[i], "ObjectType",
        "non_spandrel_beam"))
        {Flag = true;
            System.Console.WriteLine("Rule #2.: i = " + i);
        }
    }

    if (Operators.if_is_a(Element_list[i], "ElementType", "lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio(Element_list[i], "height", "width") &&
        Operators.is_dimension(Element_list[i], "aspect_ratio", ">=", 4.6) &&
        !Operators.if_is_a(Element_list[i], "ObjectType", "spandrel"))
    { if (Operators.change_element_attribute(Element_list, Element_list[i], "ObjectType",
        "spandrel"))
        { Flag = true;
            System.Console.WriteLine("Rule #3: i = " + i);
        }
    }
}

for (int j = 0; j <= Element_list.Count; j++)
{
    if (j == Element_list.Count)
    { break; }

    if (i == j) continue;
```



```

List<DB.IFCArray> elements = new List<DB.IFCArray>();
DB.IFCArray element = new DB.IFCArray();
DB.RelObj relationship = new DB.RelObj();
DB.RelObj relationship1 = new DB.RelObj();
//DB.RelObj relationship2 = new DB.RelObj();

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcColumn") &&
Operators.if_is_a( Element_list[i], "ObjectType", "split") &&
Operators.if_is_a( Element_list[j], "ObjectType", "split") &&
!Operators.if_is_a(Element_list[i], "DomainType", "checked") &&
!Operators.if_is_a(Element_list[j], "DomainType", "checked") &&
Operators.is_related_to(Rel_list, Element_list[i], Element_list[j], "split"))
{ if ( Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "column_to_column_connection", "grouted sleeve coupler or
anchor bolted connection", "column_to_column_connection",
"column_to_column_connection") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"connecting_2_columns", "connecting_2_columns_through_realizing_elements",
"connecting_2_columns") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"column_to_column_connection",
"shim_as_gravity_connection_column_to_column", "bearing_pad_C_C",
"bearing_pad_C_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.change_element_attribute(Element_list, Element_list[i], "DomainType",
"checked") &&
Operators.change_element_attribute(Element_list, Element_list[j], "DomainType",
"checked"))
{Flag=true;
System.Console.WriteLine("Rule #1: i = " + i + " j = " + j);}
}

if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
!Operators.if_is_a(Element_list[i], "ObjectType", "split") &&
(Operators.has_adjacent_faces_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) ||
Operators.has_adjacent_faces_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_narrow_faces(), 0.1)) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"connecting_beam_to_side_of_column_through_corbel"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcProjectionElement", "corbel_beam_to_column",
"corbel_connecting_beam_to_column", "corbel_beam_to_column",
"corbel_beam_to_column") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"connecting_beam_to_column",

```

```

"connecting_beam_to_side_of_column_through_corbel",
"connecting_beam_to_side_of_column_through_corbel") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"shim_connecting_beam_to_column_through_corbel",
"gravity_connection_beam_to_side_of_column",
"shim_connecting_beam_to_column_through_corbel", "bearing_pad_B_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_of_beam_stem_to_side_of_column_connection",
"tie_back_connection_beam_to_column", "tie_back_beam_to_column",
"tie_back_beam_to_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_of_beam_to_side_of_column_connection",
"tie_back_connection_beam_to_column",
"bottom_of_beam_to_side_of_column_connection", "tie_back_beam_to_column")
&&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
  System.Console.WriteLine("Rule #2.2: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
!Operators.if_is_a(Element_list[i], "ObjectType", "split") &&
Operators.has_adjacent_faces_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"connecting_beam_to_side_of_column_through_corbel"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcProjectionElement", "corbel_beam_to_column",
"corbel_connecting_beam_to_column", "corbel_beam_to_column",
"corbel_beam_to_column") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"connecting_beam_to_column",
"connecting_beam_to_side_of_column_through_corbel",
"connecting_beam_to_side_of_column_through_corbel") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"shim_connecting_beam_to_column_through_corbel",
"gravity_connection_beam_to_side_of_column",
"shim_connecting_beam_to_column_through_corbel", "bearing_pad_B_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_of_beam_stem_to_side_of_column_connection",
"tie_back_connection_beam_to_column",
"top_of_beam_stem_to_side_of_column_connection", "tie_back_beam_to_column")
&&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&

```

```

Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_of_beam_to_side_of_column_connection",
"tie_back_connection_beam_to_column",
"bottom_of_beam_to_side_of_column_connection", "tie_back_beam_to_column")
&&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
{Flag = true;
System.Console.WriteLine("Rule #2.1: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
Operators.if_is_a(Element_list[i], "ObjectType", "split") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_top_face(),
Element_list[j].horizontal_bottom_face(), 0.1) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"connecting_beam_stem_top_to_top_of_column"))
{if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "beam_stem_top_to_top_of_column_connection",
"gravity_plus_tie_back_anchor_bolted_connection",
"beam_stem_top_to_side_of_column_connection",
"beam_stem_top_to_top_of_column_connection") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"connecting_beam_stem_top_to_top_of_column",
"connecting_beam_stem_top_to_top_of_column",
"connecting_beam_stem_top_to_top_of_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_B_C", "gravity_connection_beam_to_top_of_column",
"bearing_pad_B_C", "bearing_pad_B_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
{Flag = true;
System.Console.WriteLine("Rule #3: i = " + i + " j = " + j);
}
}

}

}

}

if ((i + 1 == Element_list.Count) && (Flag == true))
{ i = -1; Flag = false; }
}

}

}

new Export_IFC(sPath, Element_list, Rel_list);

}
}
}
}

```

APPENDIX D

RULE SET FOR AUTOMATIC CLASSIFICATION AND DESIGN OF SPANDREL-COLUMN CONNECTIONS

```
for (int i = 0; i < Element_list.Count ; i++)
{
    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio( Element_list[i], "height","width") &&
        Operators.is_dimension( Element_list[i], "aspect_ratio","<","4.6") &&
        !Operators.if_is_a( Element_list[i], "ObjectType","non_spandrel_beam" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "ObjectType","non_spandrel_beam") &&
        Operators.change_element_attribute(Element_list, Element_list[i],
        "Description","non_spandrel_beam"))
        {Flag=true;
        System.Console.WriteLine("Rule #1.: i = " + i);}
    }

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&
        Operators.calculate_aspect_ratio( Element_list[i], "height","width") &&
        Operators.is_dimension( Element_list[i], "aspect_ratio",">=","4.6") &&
        !Operators.if_is_a( Element_list[i], "ObjectType","spandrel" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "ObjectType","spandrel") &&
        Operators.change_element_attribute(Element_list, Element_list[i],
        "Description","spandrel"))
        {Flag=true;
        System.Console.WriteLine("Rule #2.: i = " + i);}
    }

    if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
        !Operators.if_is_a( Element_list[i], "Description","non_load_bearing_spandrel") &&
        !Operators.if_is_a( Element_list[i], "Description","load_bearing_spandrel"))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Description","load_bearing_spandrel"))
        { Flag = true;
        System.Console.WriteLine("Rule #28: i = " + i);
        }
    }
}

for (int j = 0; j <= Element_list.Count; j++)
{
    if (j == Element_list.Count)
    {break;}
    if (i == j) continue;

    List<DB.IFCArray> elements = new List<DB.IFCArray>();
    List<DB.IFCArray> elements1 = new List<DB.IFCArray>();
}
```

```

List<DB.IFCArray> elements2 = new List<DB.IFCArray>();
DB.IFCArray element = new DB.IFCArray();
DB.RelObj relationship = new DB.RelObj();
DB.RelObj relationship1 = new DB.RelObj();
DB.RelObj relationship2 = new DB.RelObj();
DB.RelObj relationship3 = new DB.RelObj();
DB.RelObj relationship4 = new DB.RelObj();
List<DB.RelObj> relationship_list2 = new List<DB.RelObj>();
List<DB.RelObj> relationship_list1 = new List<DB.RelObj>();

if ( Operators.if_is_a( Element_list[i], "ElementType", "IfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType", "IfcSlab") &&
Operators.is_not_related_to(Rel_list, Element_list[i], Element_list[j],
"overlapping_column_slab") &&
Operators.is_overlapping( Element_list[j], Element_list[i] ))
{ if ( Operators.create_rel(Rel_list, "IfcRelAssociatesClassification", Element_list[j],
Element_list[i], "overlapping_column_slab", "overlapping_column_slab",
"overlapping_column_slab"))
{Flag=true;
System.Console.WriteLine("Rule #3: i = " + i + " j = " + j);}
}

if ( Operators.if_is_a( Element_list[i], "ElementType", "IfcColumn") &&
Operators.if_is_a( Element_list[j], "ElementType", "IfcSlab") &&
Operators.is_not_related_to(Rel_list, Element_list[i], Element_list[j],
"adjacent_column_slab") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1 ))
{ if ( Operators.create_rel(Rel_list, "IfcRelAssociatesClassification", Element_list[j],
Element_list[i], "adjacent_column_slab", "adjacent_column_slab",
"adjacent_column_slab"))
{Flag=true;
System.Console.WriteLine("Rule #4: i = " + i + " j = " + j);}
}

/* used in rule 14 and 18 */
if (Operators.if_is_a(Element_list[i], "ElementType", "IfcColumn") &&
Operators.if_is_a(Element_list[j], "ElementType", "IfcSlab") &&
(Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) ||
Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1)) &&
(Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.1) ||
Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.1)) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"aligned_column_slab_both_sides"))
{ if (Operators.create_rel(Rel_list, "IfcRelAssociatesClassification", Element_list[j],
Element_list[i], "aligned_column_slab_both_sides",
"aligned_column_slab_both_sides", "aligned_column_slab_both_sides"))
{
Flag = true;
System.Console.WriteLine("Rule #5: i = " + i + " j = " + j);}
}

```

```

/* used in rule 12, 16, and 17 */
if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcSlab") &&
(((Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) ||
Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1)) &&
!Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.1) &&
!Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.1) ||
((Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.1) ||
Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.1)) &&
!Operators.are_aligned_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) &&
!Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1))) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"aligned_column_slab_only_in_one_side"))
{if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "aligned_column_slab_only_in_one_side",
"aligned_column_slab_only_in_one_side",
"aligned_column_slab_only_in_one_side"))
{ Flag = true;
System.Console.WriteLine("Rule #6: i = " + i + " j = " + j);}
}

/* used in rule 16 and 18 */
if (Operators.if_is_a( Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcSlab") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
(Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) ||
Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.1)) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"aligned_spandrel_slab_both_sides"))
{ if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "aligned_spandrel_slab_both_sides",
"aligned_spandrel_slab_both_sides", "aligned_spandrel_slab_both_sides"))
{ Flag = true;
System.Console.WriteLine("Rule #7: i = " + i + " j = " + j);}
}

/* used in rule 11, 12, 14, 15, 17, 19, 21, 23, 26 */
if ( Operators.if_is_a( Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcSlab") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1) &&
!Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.1) &&
!Operators.are_aligned_with(Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_narrow_faces(), 0.1) &&

```

```

Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "aligned_spandrel_slab_only_in_one_side"))
{if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
    Element_list[i], "aligned_spandrel_slab_only_in_one_side",
    "aligned_spandrel_slab_only_in_one_side",
    "aligned_spandrel_slab_only_in_one_side"))
    { Flag = true;
      System.Console.WriteLine("Rule #8: i = " + i + " j = " + j);}
}

/* used in rule 21*/
if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
    Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
    Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
    Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "adjacent_spandrel_beam"))
{if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
    Element_list[i], "adjacent_spandrel_beam", "adjacent_spandrel_beam",
    "adjacent_spandrel_beam"))
    { Flag = true;
      System.Console.WriteLine("Rule #9: i = " + i + " j = " + j);}
}

if (Operators.if_is_a( Element_list[i], "ObjectType", "spandrel") &&
    Operators.if_is_a( Element_list[j], "ObjectType", "spandrel") &&
    Operators.has_adjacent_faces_with( Element_list[i].vertical_narrow_faces(),
    Element_list[j].vertical_narrow_faces(), 0.2) &&
    Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "adjacent_spandrel_spandrel_VNF_VNF"))
{if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
    Element_list[i], "adjacent_spandrel_spandrel_VNF_VNF",
    "adjacent_spandrel_spandrel_VNF_VNF",
    "adjacent_spandrel_spandrel_VNF_VNF"))
    {Flag = true;
      System.Console.WriteLine("Rule #9.1: i = " + i + " j = " + j);}
}

/* used in rule 11*/
if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
    Operators.if_is_a(Element_list[j], "ElementType", "lfcSlab") &&
    Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
    Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "adjacent_spandrel_slab"))
{if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
    Element_list[i], "adjacent_spandrel_slab", "adjacent_spandrel_slab",
    "adjacent_spandrel_slab"))
    { Flag = true;
      System.Console.WriteLine("Rule #9.2: i = " + i + " j = " + j);}
}

/* used in rule 27 for lb/nlb*/
if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
    Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
    (Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) ||
    Operators.is_overlapping( Element_list[j], Element_list[i] )) &&

```

```

Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"adjacent_or_overlapping_spandrel_column")
{ if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "adjacent_or_overlapping_spandrel_column",
"adjacent_or_overlapping_spandrel_column",
"adjacent_or_overlapping_spandrel_column"))
{Flag = true;
System.Console.WriteLine("Rule #9.3: i = " + i + " j = " + j);}
}

/* used in rule 19, 23, 27 */
if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"adjacent_column_beam"))
{ if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[j],
Element_list[i], "adjacent_column_beam", "adjacent_column_beam",
"adjacent_column_beam"))
{ Flag = true;
System.Console.WriteLine("Rule #10: i = " + i + " j = " + j); }
}

/* used in rule 23 */
if (Operators.if_is_a(Element_list[i], "ElementType", "lfcColumn") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
!Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"non_adjacent_column_beam"))
{ if (Operators.create_rel(Rel_list, "lfcRelAssociatesClassification", Element_list[i],
Element_list[j], "non_adjacent_column_beam", "non_adjacent_column_beam",
"non_adjacent_column_beam"))
{ Flag = true;
System.Console.WriteLine("Rule #10.1: i = " + i + " j = " + j); }
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"overlapping_column_slab") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"outboard_spandrel_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "bearing_pad_SP_C",

```



```

"gravity_connection_spandrel_to_pocketed_column",
"shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"outboard_spandrel_pocketed_column", "outboard_spandrel_pocketed_column",
"outboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
{Flag = true;
System.Console.WriteLine("Rule #11: i = " + i + " j = " + j);
}
}

```

```

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
!Operators.is_part_of(Rel_list, ref relationship3, Element_list[i],
"adjacent_spandrel_spandrel_VNF_VNF") &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
"aligned_spandrel_slab_only_in_one_side") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"aligned_column_slab_only_in_one_side") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"outboard_spandrel_pocketed_column"))
{if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "bearing_pad_SP_C",
"gravity_connection_spandrel_to_pocketed_column",
"shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"outboard_spandrel_pocketed_column", "outboard_spandrel_pocketed_column",
"outboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column",

```

```

"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
    System.Console.WriteLine("Rule #12: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[j],
"aligned_column_slab_both_sides") &&
Operators.belongs_to(Rel_list, ref relationship1, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship, ref elements) &&
Operators.get_related_objects(Element_list, relationship1, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "bearing_pad_SP_C",
"gravity_connection_spandrel_to_pocketed_column",
"shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column", "inboard_spandrel_pocketed_column",
"inboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_spandrel_to_pocketed_column",
"top_tie_back_spandrel_to_pocketed_column",
"top_tie_back_spandrel_to_pocketed_column",
"top_tie_back_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_spandrel_to_pocketed_column",
"bottom_tie_back_spandrel_to_pocketed_column",
"bottom_tie_back_spandrel_to_pocketed_column",
"bottom_tie_back_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
    System.Console.WriteLine("Rule #14: i = " + i + " j = " + j);
}
}
}

```

```

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1 ) &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"overlapping_column_slab") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"outboard_spandrel_ended_in_non_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcProjectionElement", "corbel_for_spandrel_to_column_connection",
"corbel_connecting_spandrel_to_column", "corbel_spandrel_to_column",
"corbel_spandrel_to_column") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"outboard_spandrel_ended_in_non_pocketed_column",
"outboard_spandrel_ended_in_non_pocketed_column",
"outboard_spandrel_ended_in_non_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_SP_C", "gravity_connection_spandrel_to_non_pocketed_column",
"shim_connecting_spandrel_to_non_pocketed_column", "bearing_pad_SP_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
System.Console.WriteLine("Rule #15: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1 ) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
"aligned_spandrel_slab_both_sides") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"aligned_column_slab_only_in_one_side") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&

```

```

Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"outboard_spandrel_ended_in_non_pocketed_column")
{ if (Operators.create_new_element(Element_list, ref element,
"lfcProjectionElement", "corbel_for_spandrel_to_column_connection",
"corbel_connecting_spandrel_to_column", "corbel_spandrel_to_column",
"corbel_spandrel_to_column") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"outboard_spandrel_ended_in_non_pocketed_column",
"outboard_spandrel_ended_in_non_pocketed_column",
"outboard_spandrel_ended_in_non_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_SP_C", "gravity_connection_spandrel_to_non_pocketed_column",
"shim_connecting_spandrel_to_non_pocketed_column", "bearing_pad_SP_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column",
"top_tie_back_outboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{Flag = true;
System.Console.WriteLine("Rule #16: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1 ) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[j],
"adjacent_column_slab") &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
"aligned_spandrel_slab_only_in_one_side") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"aligned_column_slab_only_in_one_side") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"inboard_spandrel_ended_in_non_pocketed_column"))

```

```

{ if (Operators.create_new_element(Element_list, ref element,
    "lfcProjectionElement", "corbel_for_spandrel_to_column_connection",
    "corbel_connecting_spandrel_to_column", "corbel_spandrel_to_column",
    "corbel_spandrel_to_column") &&
    Operators.create_set_from_element(elements2, element) &&
    Operators.create_rel(Rel_list, elements2, ref relationship,
    "lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
    "inboard_spandrel_ended_in_non_pocketed_column",
    "inboard_spandrel_ended_in_non_pocketed_column",
    "inboard_spandrel_ended_in_non_pocketed_column") &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bearing_pad_SP_C", "gravity_connection_spandrel_to_non_pocketed_column",
    "shim_connecting_spandrel_to_non_pocketed_column", "bearing_pad_SP_C") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{Flag = true;
    System.Console.WriteLine("Rule #17: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
    Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
    Operators.is_adjacent_to( Element_list[i], Element_list[j], 0.1 ) &&
    Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
    "aligned_spandrel_slab_both_sides") &&
    Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
    "aligned_column_slab_both_sides") &&
    Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
    "adjacent_spandrel_slab") &&
    Operators.get_related_objects(Element_list, relationship1, ref elements) &&
    Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
    Operators.compare_elements_attributes_in_lists(elements, "p21line", "=", elements1,
    "p21line") &&
    Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "inboard_spandrel_ended_in_non_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
    "lfcProjectionElement", "corbel_for_spandrel_to_column_connection",
    "corbel_connecting_spandrel_to_column", "corbel_spandrel_to_column",
    "corbel_spandrel_to_column") &&
    Operators.create_set_from_element(elements2, element) &&
    Operators.create_rel(Rel_list, elements2, ref relationship,
    "lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],

```

```

    "inboard_spandrel_ended_in_non_pocketed_column",
    "inboard_spandrel_ended_in_non_pocketed_column",
    "inboard_spandrel_ended_in_non_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bearing_pad_SP_C", "gravity_connection_spandrel_to_non_pocketed_column",
    "shim_connecting_spandrel_to_non_pocketed_column", "bearing_pad_SP_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column",
    "top_tie_back_inboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
    "bottom_tie_back_inboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
{ Flag = true;
    System.Console.WriteLine("Rule #18: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[j],
    "aligned_column_slab_only_in_one_side") &&
!Operators.is_part_of(Rel_list, ref relationship3, Element_list[i],
    "adjacent_spandrel_beam") &&
Operators.is_part_of(Rel_list, ref relationship4, Element_list[j],
    "adjacent_column_beam") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
    "adjacent_column_slab") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
    "adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
    elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "outboard_spandrel_pocketed_column"))
{if (Operators.create_new_element(Element_list, ref element,
    "lfcDiscreteAccessory", "bearing_pad_SP_C",
    "gravity_connection_spandrel_to_pocketed_column",
    "shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
    "lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[j],
    "outboard_spandrel_pocketed_column", "outboard_spandrel_pocketed_column",
    "outboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "top_tie_back_outboard_spandrel_to_pocketed_column",

```

```

"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column",
"top_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column",
"bottom_tie_back_outboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
System.Console.WriteLine("Rule #19: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
"adjacent_spandrel_beam") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"aligned_column_slab_both_sides") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&
Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "bearing_pad_SP_C",
"gravity_connection_spandrel_to_pocketed_column",
"shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column", "inboard_spandrel_pocketed_column",
"inboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{Flag = true;
System.Console.WriteLine("Rule #20: i = " + i + " j = " + j);
}
}

```



```

}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_overlapping(Element_list[i], Element_list[j]) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[j],
"non_adjacent_column_beam") &&
Operators.is_part_of(Rel_list, ref relationship3, Element_list[i],
"adjacent_spandrel_spandrel_VNF_VNF") &&
Operators.find_relationships_containing_element(Rel_list, Element_list[j],
"adjacent_column_slab", "RelatedObjects", relationship_list1) &&
Operators.find_relationships_containing_element(Rel_list, Element_list[i],
"adjacent_spandrel_slab", "RelatedObjects", relationship_list2) &&
Operators.get_related_objects(Element_list, relationship_list1, ref elements1) &&
Operators.get_related_objects(Element_list, relationship_list2, ref elements2) &&
Operators.compare_elements_attributes_in_lists(elements1, "p21line", "=",
elements2, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "bearing_pad_SP_C",
"gravity_connection_spandrel_to_pocketed_column",
"shim_connecting_spandrel_to_pocketed_column", "bearing_pad_SP_C") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"inboard_spandrel_pocketed_column", "inboard_spandrel_pocketed_column",
"inboard_spandrel_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column",
"top_tie_back_inboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
System.Console.WriteLine("Rule #22: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcColumn") &&
Operators.is_adjacent_to(Element_list[i], Element_list[j], 0.1) &&
Operators.is_part_of(Rel_list, ref relationship, Element_list[i],
"aligned_spandrel_slab_only_in_one_side") &&
Operators.is_part_of(Rel_list, ref relationship1, Element_list[j],
"adjacent_column_slab") &&
Operators.belongs_to(Rel_list, ref relationship2, Element_list[i],
"adjacent_spandrel_slab") &&
Operators.get_related_objects(Element_list, relationship1, ref elements) &&

```



```

Operators.get_related_objects(Element_list, relationship2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"inboard_spandrel_ended_in_non_pocketed_column"))
{ if (Operators.create_new_element(Element_list, ref element,
"lfcProjectionElement", "corbel_for_spandrel_to_column_connection",
"corbel_connecting_spandrel_to_column", "corbel_spandrel_to_column",
"corbel_spandrel_to_column") &&
Operators.create_set_from_element(elements2, element) &&
Operators.create_rel(Rel_list, elements2, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"inboard_spandrel_ended_in_non_pocketed_column",
"inboard_spandrel_ended_in_non_pocketed_column",
"inboard_spandrel_ended_in_non_pocketed_column") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_SP_C", "gravity_connection_spandrel_to_non_pocketed_column",
"shim_connecting_spandrel_to_non_pocketed_column", "bearing_pad_SP_C") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"top_tie_back_inboard_spandrel_to_non_pocketed_column",
"top_tie_back_inboard_spandrel_to_non_pocketed_column",
"top_tie_back_inboard_spandrel_to_non_pocketed_column",
"top_tie_back_inboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_non_pocketed_column",
"bottom_tie_back_inboard_spandrel_to_non_pocketed_column") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{ Flag = true;
System.Console.WriteLine("Rule #24: i = " + i + " j = " + j);
}
}

if (Operators.if_is_a(Element_list[i], "ObjectType", "spandrel") &&
Operators.if_is_a(Element_list[j], "ObjectType", "non_spandrel_beam") &&
Operators.find_relationships_containing_element(Rel_list, Element_list[i],
"adjacent_or_overlapping_spandrel_column", "RelatedObjects", relationship_list1)
&&
Operators.find_relationships_containing_element(Rel_list, Element_list[j],
"adjacent_column_beam", "RelatedObjects", relationship_list2) &&
Operators.get_related_objects(Element_list, relationship_list1, ref elements) &&
Operators.get_related_objects(Element_list, relationship_list2, ref elements1) &&
Operators.compare_elements_attributes_in_lists(elements, "p21line", "=",
elements1, "p21line") &&
!Operators.if_is_a( Element_list[i], "Description", "non_load_bearing_spandrel"))
{ if ( Operators.change_element_attribute(Element_list, Element_list[i],
"Description", "non_load_bearing_spandrel"))
{ Flag = true;
System.Console.WriteLine("Rule #27: i = " + i + " j = " + j);
}
}
}

```

```
    }/j
    if ((i + 1 == Element_list.Count) && (Flag == true))
    {i = -1; Flag = false;}
    }/i
    new Export_IFC(sPath, Element_list, Rel_list);
  }/main
}
}
```

APPENDIX E

RULE SET FOR AUTOMATIC DESIGN OF CONNECTIONS BETWEEN DOUBLE-TEE SLABS, SHEAR WALLS AND BEAMS

```
for (int i = 0; i < Element_list.Count ; i++)
{

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&

        Operators.is_dimension( Element_list[i], "height", ">=", 2) &&
        Operators.is_dimension( Element_list[i], "height", "<=", 5) &&
        Operators.is_dimension( Element_list[i], "width", ">", 7) &&
        !Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag", "double_tee_slab"))
    {Flag=true;
    System.Console.WriteLine("Rule #7: i = " + i );}
    }

    if ( Operators.if_is_a( Element_list[i], "ElementType","lfcBeam") &&
        Operators.is_made_of(Rel_list, Element_list[i], "concrete") &&

        Operators.is_dimension( Element_list[i], "height", ">=", 1.5) &&
        Operators.is_dimension( Element_list[i], "height", "<=", 5) &&
        Operators.is_dimension( Element_list[i], "width", ">=", 2.3) &&
        Operators.is_dimension( Element_list[i], "width", "<=", 3.4) &&
        !Operators.if_is_a( Element_list[i], "Tag", "inverted_tee_beam" ))
    { if ( Operators.change_element_attribute(Element_list, Element_list[i],
        "Tag", "inverted_tee_beam"))
    {Flag=true;
    System.Console.WriteLine("Rule #8: i = " + i );}
    }

    for (int j = 0; j <= Element_list.Count; j++)
    {
        if (j == Element_list.Count)
        {break;}
    }
    if (i == j) continue;

    List<DB.IFCArray> elements = new List<DB.IFCArray>();
    List<DB.IFCArray> elements1 = new List<DB.IFCArray>();

    List<DB.IFCArray> elements2 = new List<DB.IFCArray>();
    DB.IFCArray element = new DB.IFCArray();
    DB.RelObj relationship = new DB.RelObj();
    DB.RelObj relationship1 = new DB.RelObj();
```

```

DB.RelObj relationship2 = new DB.RelObj();
DB.RelObj relationship3 = new DB.RelObj();

List<DB.RelObj> list = new List<DB.RelObj>();
List<DB.RelObj> list1 = new List<DB.RelObj>();

float y = 0;

/* shear wall to shear wall connection type E */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 18) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 26) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_E"))
{ if ( Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)
Math.Ceiling(y/6) , "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_E_connection_relationship",
"shear_wall_to_shear_wall_type_E_connection_relationship",
"SW_to_SW_conn_rel_E"))
{Flag=true;
System.Console.WriteLine("Rule #1: i = " + i + " j = " + j);}
}

/* shear wall to shear wall connection type F */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 18) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 26) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_F"))
{ if (
Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)Math.Ceiling(y /
24), "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_grouted_coupler_connection",
"SW_to_SW_V_pocket_angle_connection",
"SW_to_SW_V_pocket_angle_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_F_connection_relationship",

```

```

"shear_wall_to_shear_wall_type_F_connection_relationship",
"SW_to_SW_conn_rel_F"))
{Flag=true;
System.Console.WriteLine("Rule #2: i = " + i + " j = " + j);}
}

/* shear wall to shear wall connection type E */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 26) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 36) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_E"))
{ if (Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)
Math.Ceiling(y/6), "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_E_connection_relationship",
"shear_wall_to_shear_wall_type_E_connection_relationship",
"SW_to_SW_conn_rel_E"))

{Flag=true;
System.Console.WriteLine("Rule #1.1: i = " + i + " j = " + j);}
}

/* shear wall to shear wall connection type F */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 26) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 36) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_F"))
{ if (Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)Math.Ceiling(y /
24), "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_grouted_coupler_connection",
"SW_to_SW_V_pocket_angle_connection",
"SW_to_SW_V_pocket_angle_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_F_connection_relationship",
"shear_wall_to_shear_wall_type_F_connection_relationship",
"SW_to_SW_conn_rel_F"))
{Flag=true;
System.Console.WriteLine("Rule #2.1: i = " + i + " j = " + j);}
}

```

```

}

/* shear wall to shear wall connection type E */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 8) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 18) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_E"))
{ if (Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)
Math.Ceiling(y/6), "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection",
"SW_to_SW_mechanical_anchor_bolted_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_E_connection_relationship",
"shear_wall_to_shear_wall_type_E_connection_relationship",
"SW_to_SW_conn_rel_E"))

{Flag=true;
System.Console.WriteLine("Rule #1.2: i = " + i + " j = " + j);}
}

/* shear wall to shear wall connection type F */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.has_adjacent_faces_with(Element_list[i].horizontal_bottom_face(),
Element_list[j].horizontal_top_face(), 0.1) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", ">", 8) &&
Operators.is_dimension( Element_list[j], "Top_Elevation", "<=", 18) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"SW_to_SW_conn_rel_F"))
{ if (Operators.get_element_dimension(Element_list[i], "length", ref y) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)Math.Ceiling(y /
24), "lfcDiscreteAccessory", "shear_wall_to_shear_wall_connection",
"SW_to_SW_grouted_coupler_connection",
"SW_to_SW_V_pocket_angle_connection",
"SW_to_SW_V_pocket_angle_connection") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"shear_wall_to_shear_wall_type_F_connection_relationship",
"shear_wall_to_shear_wall_type_F_connection_relationship",
"SW_to_SW_conn_rel_F"))

{Flag=true;
System.Console.WriteLine("Rule #2.2: i = " + i + " j = " + j);}
}

```

```

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcBeam") &&
    Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
    Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab") &&
    Operators.if_is_a( Element_list[i], "Name", "CEG_DT:12DT30") &&
    Operators.has_adjacent_faces_with( Element_list[i].vertical_narrow_faces(),
    Element_list[j].vertical_wide_faces(), 0.2) &&
    Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
    "perpendicular_DT_to_SW_connection")&&
    Operators.compare_elements_attribute(Element_list[j], "Bottom_Elevation", "<=",
    Element_list[i], "Bottom_Elevation"))
{ if (
    Operators.create_new_element(Element_list, ref element, "lfcProjectionElement",
    "corbel_for_double_tee_to_shear_wall_connection",
    "corbel_1_connecting_perpendicular_DT_to_SW", "corbel_1_DT_to_SW",
    "corbel_1_DT_to_SW") &&
    Operators.create_set_from_element(elements, element) &&
    Operators.create_rel(Rel_list, elements, ref relationship,
    "lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
    "double_tee_to_shear_wall_connection_relationship",
    "perpendicular_DT_to_SW_connection", "perpendicular_DT_to_SW_connection") &&
    Operators.create_new_element(Element_list, ref element, "lfcProjectionElement",
    "corbel_for_double_tee_to_shear_wall_connection",
    "corbel_2_connecting_perpendicular_DT_to_SW", "corbel_2_DT_to_SW",
    "corbel_2_DT_to_SW") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "perpendicular_double_tee_to_shear_wall_connection",
    "DT_to_SW_fixed_welded_connection",
    "DT_to_SW_slotted_insert_wall_embed_plate_flange",
    "DT_to_SW_slotted_insert_wall_embed_plate_flange1") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "perpendicular_double_tee_to_shear_wall_connection",
    "DT_to_SW_fixed_welded_connection",
    "DT_to_SW_slotted_insert_wall_embed_plate_flange2",
    "DT_to_SW_slotted_insert_wall_embed_plate_flange") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bearing_pad_DT_SW",
    "gravity_connection_double_tee_to_shear_wall_connection",
    "shim_connecting_DT_to_SW_with_corbel", "bearing_pad_DT_SW1") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
    "bearing_pad_DT_SW",
    "gravity_connection_double_tee_to_shear_wall_connection",
    "shim_connecting_DT_to_SW_with_corbel", "bearing_pad_DT_SW2") &&
    Operators.add_object_to_relationship(relationship, element, "RealizingElements")
    &&
    Operators.create_new_element(Element_list, ref element, "lfcProjectionElement",
    "corbel_for_double_tee_to_shear_wall_connection",

```

```

"corbel_1_connecting_perpendicular_DT_to_SW", "corbel_1_DT_to_SW",
"corbel_1_DT_to_SW") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"double_tee_to_shear_wall_connection_relationship",
"perpendicular_DT_to_SW_connection", "perpendicular_DT_to_SW_connection") &&
Operators.create_new_element(Element_list, ref element, "lfcProjectionElement",
"corbel_for_double_tee_to_shear_wall_connection",
"corbel_2_connecting_perpendicular_DT_to_SW", "corbel_2_DT_to_SW",
"corbel_2_DT_to_SW") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"perpendicular_double_tee_to_shear_wall_connection",
"DT_to_SW_fixed_welded_connection",
"DT_to_SW_slotted_insert_wall_embed_plate_flange",
"DT_to_SW_slotted_insert_wall_embed_plate_flange1") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"perpendicular_double_tee_to_shear_wall_connection",
"DT_to_SW_fixed_welded_connection",
"DT_to_SW_slotted_insert_wall_embed_plate_flange2",
"DT_to_SW_slotted_insert_wall_embed_plate_flange") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_DT_SW",
"gravity_connection_double_tee_to_shear_wall_connection",
"shim_connecting_DT_to_SW_with_corbel", "bearing_pad_DT_SW1") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_DT_SW",
"gravity_connection_double_tee_to_shear_wall_connection",
"shim_connecting_DT_to_SW_with_corbel", "bearing_pad_DT_SW2") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{
    Flag = true;
    System.Console.WriteLine("Rule #3: i = " + i + " j = " + j);}
}

/* shear wall to parallel intersecting Double Tee connections */

if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcBeam") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcWallStandardCase") &&
Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab") &&
Operators.if_is_a( Element_list[i], "Name", "CEG_DT:12DT30") &&
(Operators.has_adjacent_faces_with( Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.2) ||
Operators.is_overlapping( Element_list[j], Element_list[i] )) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"parallel_DT_to_SW_connection"))
{ if ((Operators.elements_overlapping(Element_list[i], Element_list[j], "length", ref y) ||

```



```

Operators.elements_adjacency( Element_list[i], Element_list[j], "length", ref y)) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)Math.Ceiling(y /
2), "lfcDiscreteAccessory", "parallel_double_tee_to_shear_wall_connection",
"DT_to_SW_adjustable_connection",
"DT_to_SW_slotted_insert_wall_embed_plate_flange",
"DT_to_SW_slotted_insert_wall_embed_plate_flange") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"double_tee_to_shear_wall_connection_relationship",
"parallel_DT_to_SW_connection", "parallel_DT_to_SW_connection"))
{Flag = true;
System.Console.WriteLine("Rule #4: i = " + i + " j = " + j);}
}

/* Double Tee to Double Tee connections */
if (Operators.if_is_a(Element_list[i], "ElementType", "lfcBeam") &&
Operators.if_is_a(Element_list[j], "ElementType", "lfcBeam") &&
Operators.if_is_a(Element_list[i], "Tag", "double_tee_slab") &&
Operators.if_is_a(Element_list[j], "Tag", "double_tee_slab") &&
Operators.if_is_a(Element_list[i], "Name", "CEG_DT:12DT30") &&
Operators.if_is_a(Element_list[j], "Name", "CEG_DT:12DT30") &&
(Operators.has_adjacent_faces_with(Element_list[i].vertical_wide_faces(),
Element_list[j].vertical_wide_faces(), 0.2) ||
Operators.is_overlapping(Element_list[i], Element_list[j]))&&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"DT_to_DT_connection"))
{if (
(Operators.elements_adjacency(Element_list[i], Element_list[j], "length", ref y) ||
Operators.elements_overlapping( Element_list[i], Element_list[j], "length", ref y)) &&
Operators.create_list_of_new_elements(Element_list, elements, (int)Math.Ceiling(y /
7), "lfcDiscreteAccessory", "double_tee_to_double_tee_flange_connection",
"DT_to_DT_welded_connection_with_embed_plates",
"DT_to_DT_embed_plates_in_flange_and_slug",
"DT_to_DT_embed_plates_in_flange_and_slug") &&
Operators.create_list_of_relationships(Rel_list, elements,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"double_tee_to_double_tee_connection_relationship", "DT_to_DT_connection",
"DT_to_DT_connection"))
{Flag = true;
System.Console.WriteLine("Rule #5: i = " + i + " j = " + j);}
}

/* Double Tee to Beam connections */
if ( Operators.if_is_a( Element_list[i], "ElementType", "lfcBeam") &&
Operators.if_is_a( Element_list[j], "ElementType", "lfcBeam") &&
Operators.if_is_a( Element_list[i], "Tag", "double_tee_slab") &&
Operators.if_is_a( Element_list[j], "Name", "CEG_DT:12DT30") &&
Operators.if_is_a( Element_list[j], "Name", "CEG_IT_Beam") &&
Operators.if_is_a(Element_list[j], "Tag", "inverted_tee_beam") &&
(Operators.has_adjacent_faces_with( Element_list[i].vertical_narrow_faces(),
Element_list[j].vertical_wide_faces(), 0.2) ||
Operators.is_overlapping( Element_list[j], Element_list[i] )) &&
Operators.is_not_related_to(Rel_list, Element_list[j], Element_list[i],
"DT_to_IT_or_L_beam_connection"))

```

```

{ if (Operators.create_new_element(Element_list, ref element,
"lfcDiscreteAccessory", "double_tee_to_beam_connection",
"DT_to_IT_or_L_beam_welded_connection",
"embed_angle_in_DT_flange_and_embed_plate_in_ITB",
"embed_angle_in_DT_flange_and_embed_plate_in_ITB") &&
Operators.create_set_from_element(elements, element) &&
Operators.create_rel(Rel_list, elements, ref relationship,
"lfcRelConnectsWithRealizingElements", Element_list[j], Element_list[i],
"double_tee_to_beam_connection_relationship",
"DT_to_IT_or_L_beam_connection", "DT_to_IT_or_L_beam_connection") &&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_DT_IT_or_L_beam",
"gravity_connection_double_tee_to_inveted_tee_or_L_beam",
"shim_connecting_DT_to_IT_or_L_beam", "bearing_pad_DT_IT_or_L_beam1") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements")
&&
Operators.create_new_element(Element_list, ref element, "lfcDiscreteAccessory",
"bearing_pad_DT_IT_or_L_beam",
"gravity_connection_double_tee_to_inveted_tee_or_L_beam",
"shim_connecting_DT_to_IT_or_L_beam", "bearing_pad_DT_IT_or_L_beam2") &&
Operators.add_object_to_relationship(relationship, element, "RealizingElements"))
{
    Flag = true;
    System.Console.WriteLine("Rule #6: i = " + i + " j = " + j);}
}

```

REFERENCES

- [1] Staub-French, S., Fischer, M., Kunz, J., & Paulson, B. A. (2003). Generic feature-driven activity-based cost estimation process. *Advanced Engineering Informatics*, 17(1), 23-39.
- [2] Sacks, R., & Barak, R. (2008). Impact of three-dimensional parametric modeling of buildings on productivity in structural engineering practice. *Automation in Construction*, 17(4), 439-449.
- [3] Sacks, R., Eastman, C. M., Lee, G., & Orndorff, D. (2005). A target benchmark of the impact of three-dimensional parametric modeling in precast construction. *PCI Journal*, 50(4), 126-139.
- [4] Aram, S., Eastman, C., & Sacks, R. (2013). Requirements for BIM platforms in the concrete reinforcement supply chain. *Automation in Construction*, 35, 1-17.
- [5] Hartmann, T., Gao, J., & Fischer, M. (2008). Areas of application for 3D and 4D models on construction projects. *Journal of Construction Engineering and Management*, 134(10), 776-785.
- [6] Forgues, D., Iordanova, I., Valdivesio, F., & Staub-French, S. (2012). Rethinking the cost estimating process through 5D BIM: A case study. *Construction Research Congress* 778-786.
- [7] Sacks, R., Eastman, C. M., & Lee, G. (2004). Process model perspectives on management and engineering procedures in the precast/prestressed concrete industry. *Journal of Construction Engineering and Management*, 130(2), 206-215.
- [8] McGraw Hill Construction SmartMarket Report. (2012). *The Business Value of BIM in North America: Multi-Year trend analysis and user ratings (2007-2012)*. McGraw-Hill Construction, New York.
- [9] AACE International (1997). *Recommended practice: cost estimate classification system – as applied in engineering, procurement and construction for the process industries*. SSVR 18R-97, Morgantown, WV.

- [10] Cavalieri, S., Maccarrone, P., Pinto, R. (2004). Parametric vs. neural network models for the estimation of production costs: A case study in the automotive industry. *International Journal of Production Economic*, 91(2),165–177.
- [11] Niazi, A., Dai, J.S., Balabani, S., & Seneviratne, L. (2006). Product cost estimation: technique classification and methodology review. *Journal of Manufacturing Science and Engineering*, 128 (2), 563–575.
- [12] Chougule, R. G., & Ravi, B. (2006). Casting cost estimation in an integrated product and process design environment. *International Journal of Computer Integrated Manufacturing*, 19(7), 676-688.
- [13] Bode, J. (2000). Neural networks for cost estimation: simulations and pilot application. *International Journal of Production Research*, 38(6), 1231-1254.
- [14] Kim, G. H., An, S.H., & Kang, K. I. (2004). Comparison of construction cost estimating models based on regression analysis, neural networks, and case-based reasoning. *Building & Environment*, 39, 1235–1242.
- [15] Arditi, D., & Tokdemir, O. B. (1999). Comparison of case-based reasoning and artificial neural networks. *Journal of computing in civil engineering*, 13(3), 162-169.
- [16] Hegazy, T., & Ayed, A. (1998). Neural network model for parametric cost estimation of highway projects. *Journal of Construction Engineering and Management*, 124(3), 210-218.
- [17] Tatari, O., & Kucukvar, M. (2011). Cost premium prediction of certified green buildings: A neural network approach. *Building and Environment*, 46(5), 1081-1086.
- [18] Koo, C., Hong, T., Hyun, C., & Koo, K. (2010). A CBR-based hybrid model for predicting a construction duration and cost based on project characteristics in multi-family housing projects. *Canadian Journal of Civil Engineering*, 37(5), 739-752.
- [19] Günaydin, H. M., & Doğan, S. Z. (2004). A neural network approach for early cost estimation of structural systems of buildings. *International Journal of Project Management*, 22, 595–602.
- [20] Kim, G. H., Yoon, J. E., An, S.H., Cho, H. H., & Kang, K. I. (2004). Neural network model incorporating a genetic algorithm in estimating construction costs. *Building & Environment*, 39, 1333–1340.

- [21] Doğan, S. Z., Arditi, D., & Murat Günaydin, H. (2008). Using decision trees for determining attribute weights in a case-based model of early cost prediction. *Journal of Construction Engineering and Management*, 134(2), 146-152.
- [22] An, S. H., Kim, G. H., & Kang, K. I. (2007). A case-based reasoning cost estimating model using experience by analytic hierarchy process. *Building and Environment*, 42(7), 2573-2579.
- [23] Jin, R., Cho, K., Hyun, C., & Son, M. (2012). MRA-based revised CBR model for cost prediction in the early stage of construction projects. *Expert Systems with Applications*, 39(5), 5214-5222.
- [24] Marzouk, M. M., & Ahmed, R. M. (2011). A case-based reasoning approach for estimating the costs of pump station projects. *Journal of Advanced Research*, 2(4), 289-295.
- [25] Kim, H. J., Seo, Y. C., & Hyun, C. T. (2012). A hybrid conceptual cost estimating model for large building projects. *Automation in Construction*, 25, 72-81.
- [26] Qian, L., & Ben-Arieh, D. (2008). Parametric cost estimation based on activity-based costing: A case study for design and development of rotational parts. *International Journal of Production Economics*, 113(2), 805-818.
- [27] H'mida, F., Martin, P., & Vernadat, F. (2006). Cost estimation in mechanical production: The Cost Entity approach applied to integrated product engineering. *International Journal of Production Economics*, 103(1), 17-35.
- [28] Roy, R., Souchoroukov, P., & Griggs, T. (2008). Function-based cost estimating. *International Journal of Production Research*, 46(10), 2621-2650.
- [29] Moharrami, H. & Grierson, D. E. (1993). Computer-automated design of reinforced concrete frameworks, *ASCE Journal of Structural Engineering*, 119(7), 2036-58.
- [30] Hartgraves, A. L., & Morse, W. M. (2012). Chapter 2: Cost behavior, activity analysis, and cost estimation. *Managerial Accounting*, 6th ed., Cambridge Business Publishers, Westmont, IL.
- [31] Aram, S., Eastman, C., Beetz, J. (2014). Qualitative and Quantitative Cost Estimation: A Methodology Analysis, ICCCB E 2014 & 2014 CIB W78, Orlando, Florida.

- [32] Renner, G., & Ekárt, A. (2003). Genetic algorithms in computer aided design. *Computer-Aided Design*, 35(8), 709-726.
- [33] Kicing, R., Arciszewski, T., & Jong, K. D. (2005). Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23), 1943-1978.
- [34] Koumoussis, V. K., & Georgiou, P. G. (1994). Genetic algorithms in discrete optimization of steel truss roofs. *Journal of Computing in Civil Engineering*, 8(3), 309-325.
- [35] Rajeev, S., & Krishnamoorthy, C. S. (1992). Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5), 1233-1250.
- [36] Nimitawat, A., & Nanakorn, P. (2009). Automated layout design of beam-slab floors using a genetic algorithm. *Computers & Structures*, 87(21), 1308-1330.
- [37] Pullmann, T., Skolicki, Z., Freischlad, M., Arciszewski, T., De Jong, K. A., & Schnellenbach-Held, M. (2003). Structural design of reinforced concrete tall buildings: evolutionary computation approach using fuzzy sets. Proc. of the 10th International Workshop of the European Group for Intelligent Computing in Engineering (EG-ICE). Delft, The Netherlands.
- [38] Woon, S. Y., Querin, O. M., & Steven, G. P. (2001). Structural application of a shape optimization method based on a genetic algorithm. *Structural and Multidisciplinary Optimization*, 22(1), 57-64.
- [39] Lee, K. S., & Geem, Z. W. (2004). A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9), 781-798.
- [40] Sarma, K. C., & Adeli, H. (2001). Bilevel parallel genetic algorithms for optimization of large steel structures. *Computer-Aided Civil and Infrastructure Engineering*, 16(5), 295-304.
- [41] Paya, I., Yepes, V., González-Vidos, F., & Hospitaler, A. (2008). Multi-objective optimization of concrete frames by simulated annealing. *Computer-Aided Civil and Infrastructure Engineering*, 23(8), 596-610.
- [42] Rosenman, M. A., & Gero, J. S. (1999). Evolving designs by generating useful complex gene structures. *Evolutionary Design by Computers*, 345-364.

- [43] Shea, K., Sedgwick, A., & Antonunntto, G. (2006). Multicriteria optimization of paneled building envelopes using ant colony optimization. *Intelligent Computing in Engineering and Architecture*, 627-636, Springer Berlin Heidelberg.
- [44] Krish, S. (2011). A practical generative design method. *Computer-Aided Design*, 43(1), 88-100.
- [45] O'Neill, M., McDermott, J., Swafford, J. M., Byrne, J., Hemberg, E., Brabazon, A., & Hemberg, M. (2010). Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. *International Journal of Design Engineering*, 3(1), 4-24.
- [46] Maher, M. L. (2000). A model of co-evolutionary design. *Engineering with computers*, 16(3-4), 195-208.
- [47] Hofmeyer, H., Rutten, H. S., & Fijneman, H. J. (2006). Interaction of spatial and structural design, an automated approach. *Design Studies*, 27(4), 423-438.
- [48] Singh, S. P., & Sharma, R. R. K. (2006). A review of different approaches to the facility layout problems. *The International Journal of Advanced Manufacturing Technology*, 30(5-6), 425-433.
- [49] Michalek, J., Choudhary, R., & Papalambros, P. (2002). Architectural layout design optimization. *Engineering optimization*, 34(5), 461-484.
- [50] Chapman, C. B., & Pinfold, M. (1999). Design engineering—a need to rethink the solution using knowledge based engineering. *Knowledge-based Systems*, 12(5), 257-267.
- [51] Sacks, R., Warszawski, A., & Kirsch, U. (2000). Structural design in an automated building system. *Automation in Construction*, 10(1), 181-197.
- [52] Novak, M., & Došak, B. (2008). Intelligent FEA-based design improvement. *Engineering Applications of Artificial Intelligence*, 21(8), 1239-1254.
- [53] Frank, G. (2013). An expert system for design-process automation in a CAD environment. In *ICONS 2013, The Eighth International Conference on Systems*, 179-184.
- [54] Akerkar, R. A. and Sajja Priti Srinivas (2010). *Knowledge-based systems*. Jones & Bartlett Publishers, Sudbury, MA, USA.

- [55] Dhaliwal, J. S., & Benbasat, I. (1996). The use and effects of knowledge based system explanations: theoretical foundations and a framework for empirical evaluation. *Information Systems Research*, 7(3), 342–362, 1996.
- [56] Milton, N. R. (2008). *Knowledge technologies (Vol. 3)*. Polimetrica sas.
- [57] Rittle-Johnson, B., & Siegler, R. S. (1998). The relation between conceptual and procedural knowledge in learning mathematics: A review. In C. Donlan (Ed.), *The development of mathematical skills*, 75–110, East Sussex, England: Psychology Press.
- [58] Emberey CL, Milton NR, Berends JPTJ, van Tooren MJL, van der Elst SWG, & Vermeulen B. (2007). Application of knowledge engineering methodologies to support engineering design application development in aerospace. Proc. of 7th AIAA Aviation Technology, Integration and Operations Conference (ATIO), AIAA-2007-7708, Belfast, Ireland.
- [59] Chapman, C. B., & Pinfold, M. (2001). The application of a knowledge based engineering approach to the rapid design and analysis of an automotive structure. *Advances in Engineering Software*, 32(12), 903-912.
- [60] Cooper, D., & La Rocca, G. (2007). Knowledge-based techniques for developing engineering applications in the 21st century. 7th AIAA ATIO Conference, Belfast, Northern Ireland.
- [61] Verhagen, W. J., Bermell-Garcia, P., van Dijk, R. E., & Curran, R. (2012). A critical review of Knowledge-Based Engineering: An identification of research challenges. *Advanced Engineering Informatics*, 26(1), 5-15.
- [62] Shehab, E. & Abdalla, H. (2002). An intelligent knowledge-based system for product cost modeling. *International Journal of Advanced Manufacturing Technology*, 19:49–65.
- [63] Chan D. S. K. & Lewis W.P. (2000). The integration of manufacturing and cost information into the engineering design process. *International Journal of Production Research*, 38(17), 4413-4427.
- [64] Bouaziz, Z., Younes, J. B. & Zghal, A. (2006). Cost estimation system of dies manufacturing based on the complex machining features. *The International Journal of Advanced Manufacturing Technology*, 28(3-4), 262-271.

- [65] Ko, K. H., Pochiraju, K. & Manoochehri, S. (2007). An embedded system for knowledge-based cost evaluation of molded parts. *Knowledge-Based Systems*, 20, 291–299.
- [66] Sobolewski, M. (2002). *Federated P2P Services in CE Environments*. Advances in Concurrent Engineering. A.A. Balkema Publishers, ISBN 90 5809 502 9,13–22.
- [67] Koonce, D., Judd R., Sormaz D. & Masel, D. T. (2003). A hierarchical cost estimation tool. *Computers in Industry* 50: 293–302.
- [68] Benjaoran, V., & Dawood, N. (2006). Intelligence approach to production planning system for bespoke precast concrete products. *Automation in construction*, 15(6), 737-745.
- [69] García-Crespo, Á., Ruiz-Mezcua, B., López-Cuadrado, J. L., & González-Carrasco, I. (2011). A review of conventional and knowledge based systems for machining price quotation. *Journal of Intelligent Manufacturing*, 22(6), 823-841.
- [70] Sandberg, M., Boart, P., & Larsson, T. (2005). Functional product life-cycle simulation model for cost estimation in conceptual design of jet engine components. *Concurrent Engineering*, 13(4), 331-342.
- [71] Lee, S. K., Kim, K. R. & Yu, J. H. (2013). BIM and ontology-based approach for building cost estimation. *Automation in Construction*, 41, 96-105.
- [72] Steel, J., & Drogemuller, R. (2011). Domain-specific model transformation in building quantity take-off. *Proc. of Model Driven Engineering Languages and Systems (MoDELS) International Conference, LNCS*, 198–212, Berlin/Heidelberg.
- [73] Tabatabai-Gargari, M., & Elzarka, H. M. (1998). Integrated CAD/KBS approach for automating preconstruction activities. *Journal of Construction Engineering and Management*, 124(4), 257-262.
- [74] Studer, R., Benjamins, V. R. & Fensel, D. (1998). Knowledge engineering: principles and methods. *Data & knowledge Engineering*, 25(1), 161-197.
- [75] Neches, R., Fikes, R.E., Finin, T., Gruber, T.R., Senator, T. & Swartout, W.R. (1991). Enabling technology for knowledge sharing, *AI Magazine*, 12(3) 36-56.

- [76] Li, B. M., Xie, S. Q. & Xu, X. (2011). Recent development of knowledge-based systems, methods and tools for one-of-a-kind production. *Knowledge-Based Systems*. 24(7), 1108-1119.
- [77] Fensel, D. (2002). Ontology-based knowledge management. *Computer*, 35(11), 56-59.
- [78] Kim, C. H., Weston, R. H., Hodgson, A. & Lee, K. H. (2003). The complementary use of IDEF and UML modelling approaches. *Computers in Industry*, 50(1), 35-56.
- [79] Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2) 199-220.
- [80] Schreiber, G., Wielinga, B. & Breuker, J. (1993). KADS: A principled approach to knowledge-based system development. Academic Press, 11.
- [81] Belsky, M., Sacks, R., Brilakis, I. (2014). SEEBIM: A Semantic enrichment engine for building information modeling. *Computer-Aided Civil and Infrastructure Engineering* (Accepted).
- [82] RDF Ltd, IFC Engine, <http://rdf.bg/ifc-engine-dll.php?page=products> (last accessed on 12/15/2014).
- [83] RDF Ltd, IFC Viewer <http://rdf.bg/ifc-viewer.php> (last accessed on 12/15/2014).
- [84] Danjou, S., Lupa, N., & Koehler, P. (2008). Approach for automated product modeling using knowledge-based design features. *Computer-Aided Design and Applications*, 5(5), 622-629.
- [85] ICC (2009), IBC (2009). The International Building Code (IBC). International Code Council (ICC), INC, Country Club Hill, IL 60478.
- [86] PCI Industry Handbook Committee (2010). PCI design handbook: 7th ed., Precast/Prestressed Concrete Institute, Chicago, IL 60606-5230.
- [87] PCI. A guide to designing with precast and prestressed concrete. http://www.gpcpi.org/index.cfm/precast_solutions/primer
- [88] PCI Connections Detail Committee (2008). PCI connections manual for precast and prestressed concrete construction, 1st ed., Precast/Prestressed Concrete Institute, Chicago, IL 60606-5230.

- [89] ACI Committee 318 (2005). Building code requirements for structural concrete and commentary (ACI 318-05), American Concrete Institute (ACI).
- [90] National Precast Concrete Association (NCPA) (2011). Architectural precast concrete wall panels: connection guide, <http://precast.org/wp-content/uploads/2012/01/ArchitecturalConnectionsGuide.pdf>
- [91] Hartmann, T., Van Meerveld, H., Vosseveld, N., & Adriaanse, A. (2012). Aligning building information model tools and construction management methods. *Automation in construction*, 22, 605-613.
- [92] Shim, J. P., Warkentin, M., Courtney, J. F., Power, D. J., Sharda, R., & Carlsson, C. (2002). Past, present, and future of decision support technology. *Decision Support Systems*, 33(2), 111-126.
- [93] Van der Elst, S. W. G., & Van Tooren, M. J. L. (2008). Application of a knowledge engineering process to support engineering design application development. *Collaborative Product and Service Life Cycle Management for a Sustainable World*, 417-43, Springer London.
- [94] Skarka, W. (2007). Application of MOKA methodology in generative model creation using CATIA. *Engineering Applications of Artificial Intelligence*, 20(5), 677-690.
- [95] Gómez-Pérez, A., & Benjamins, R. (1999). Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods. *IJCAI and the Scandinavian AI Societies. CEUR Workshop Proceedings*.
- [96] Aliverti, E. (2012). About Drools and infinite execution loops. <http://www.plugtree.com/about-drools-and-infinite-execution-loops/> (last accessed on 11/23/2014).
- [97] Lee, G., Sacks, R., & Eastman, C. M. (2006). Specifying parametric building object behavior (BOB) for a building information modeling system. *Automation in construction*, 15(6), 758-776.
- [98] Euzenat, J. (2001). Towards a principled approach to semantic interoperability. *IJCAI Workshop on Ontology and Information Sharing*, Seattle, USA, pp. 19–25.
- [99] Liao, Y., Lezoche, M., Loures, E., Panetto, H., & Boudjlida, N. (2013). Semantic enrichment of models to assist knowledge management in a PLM environment. In *On the Move to Meaningful Internet Systems: OTM 2013 Conferences* (pp. 267-274). Springer Berlin Heidelberg.

- [100] ISO10303-11 (1994). 'Industrial automation systems and integration -- Product data representation and exchange --Part 11: Description methods: The EXPRESS language reference manual'. International Organization for Standardization (ISO).
- [101] Liebich, T., Adachi, Y., Forester, J., Hyvarinen, J., Richter, S., Chipman, T. et al., Industry Foundation Classes, IFC 2x Edition 4 Release Candidate 2, Model Support Group (MSG) of buildingSMART <http://buildingSMART-tech.org/ifc/IFC2x4/alpha/html/> (last accessed on 11/25/2014).
- [102] Panetto, H. (2007). Towards a classification framework for interoperability of enterprise applications. *International Journal of Computer Integrated Manufacturing*, 20(8), 727-740.
- [103] Eastman, C. M., Jeong, Y. S., Sacks, R., & Kaner, I. (2009). Exchange model and exchange object concepts for implementation of national BIM standards. *Journal of Computing in Civil Engineering*, 24(1), 25-34.
- [104] Aram, S., Eastman, C., Venugopal, M., Sacks, R., & Belsky, M. (2013). Concrete reinforcement modeling for efficient information sharing. Proc. of the 30th International Symposium on Automation and Robotics in Construction (ISARC) - Montreal, Canada.
- [105] Beetz, J., van Berlo, L., de Laat, R., & Bonsma, P. (2011). Advances in the development and application of an open source model server for building information. Proc. of CIB W078-W102, Sophia Antipolis, France.
- [106] Aram, S., & Eastman, C. (2013). Integration of PLM solutions and BIM systems for the AEC industry. Proc. of the 30th International Symposium on Automation and Robotics in Construction (ISARC) - Montreal, Canada.
- [107] You, C. F., & Tsai, Y. L. (2010). 3D solid model retrieval for engineering reuse based on local feature correspondence. *The International Journal of Advanced Manufacturing Technology*, 46(5-8), 649-661.
- [108] El-Mehalawi, M., & Allen Miller, R. (2003). A database system of mechanical components based on geometric and topological similarity. Part II: indexing, retrieval, matching, and similarity assessment. *Computer-Aided Design*, 35(1), 95-105.
- [109] Tsai, C. Y., & Chang, C. A. (2005). A two-stage fuzzy approach to feature-based design retrieval. *Computers in Industry*, 56(5), 493-505.

- [110] Katranuschkov, P., Weise, M., Windisch, R., Fuchs, S., & Scherer, R. J. (2010). BIM-based generation of multi-model views. CIB W78.
- [111] Nepal, M. P., Staub-French, S., Pottinger, R., & Zhang, J. (2012). Ontology-based feature modeling for construction information extraction from a building information model. *Journal of Computing in Civil Engineering*, 27(5), 555-569.
- [112] Adachi, Y. (2003). Overview of partial model query language. In ISPE CE (pp. 549-555).
- [113] Borrmann, A., & Rank, E. (2009). Topological analysis of 3D building models using a spatial query language. *Advanced Engineering Informatics*, 23(4), 370-385.
- [114] Foucault, G., & Léon, J. C. (2010). Enriching assembly CAD models with functional and mechanical informations to ease CAE. Proc. Of ASME 2010 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (pp. 341-351). American Society of Mechanical Engineers.
- [115] Liao, Y., Lezoche, M., Loures, E., Panetto, H., & Boudjlida, N. (2013). Semantic enrichment of models to assist knowledge management in a PLM environment. Proc. of on the Move to Meaningful Internet Systems: OTM 2013 Conferences (pp. 267-274). Springer Berlin Heidelberg.
- [116] Solibri Inc., Solibri Model Viewer, <http://www.solibri.com/products/solibri-model-viewer/> (last accessed on 12/1/2014).
- [117] Eastman, C.M., Teicholz, P., Sacks, R., Liston, K. (2011). BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors, 2nd ed. Wiley.
- [118] Wilson, C., Sutter medical center Castro Valley: IPD process innovation with building information modeling, <http://network.aia.org/technologyinarchitecturalpractice/Resources> (last accessed on 10/13/2014).
- [119] Scopano, S., Allen, C., Cousins, B., Muir, W., Martino, R., Schoen, R. (2009). Interrupting the supply chain - Whitepaper, www.tekla.com/us/Documents/Interlocken_whitepaper.pdf (last accessed on 05/21/2013).