A WEB BASED MULTI-USER FRAMEWORK FOR THE DESIGN AND DETAILING OF REINFORCED CONCRETE FRAMES - COLUMNS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÖKHAN ÜNAL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

DECEMBER 2009

Approval of the thesis

"**A WEB BASED MULTI-USER FRAMEWORK FOR THE DESIGN AND DETAILING OF REINFORCED CONCRETE FRAMES - COLUMNS** "

submitted by **Gökhan Ünal** in partial fullfillment of the requirements for the degree of **Master of Science in Civil Engineering** by,

Prof. Dr. Canan Özgen _____
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Güney Özcebe _____
Head of Department, **Civil Engineering**

Assist. Prof. Dr. Özgür Kurç _____
Supervisor, **Departmant of Civil Engineering, METU**

Examining Committee Members:

Prof. Dr. Güney Özcebe _____
Departmant of Civil Engineering, METU

Assist. Prof. Dr. Özgür Kurç _____
Departmant of Civil Engineering, METU

Assoc. Prof. Dr. Erdem CANBAY _____
Departmant of Civil Engineering, METU

Assist. Prof. Dr. Ali Murat Tanyer _____
Departmant of Architecture, METU

Assist. Prof. Dr. Semiha Ergan _____
Departmant of Civil Engineering, METU

Date: _____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : Gökhan Ünal

Signature :

ABSTRACT

A WEB BASED MULTI-USER FRAMEWORK FOR THE DESIGN AND DETAILING OF REINFORCED CONCRETE FRAMES - COLUMNS

Ünal, Gökhan

M.S, Department of Civil Engineering

Supervisor: Assist. Prof. Dr. Özgür Kurç

December 2009, 93 pages

In design and detailing of a reinforced concrete frame project, there are many engineers who contribute to a single project. Wide variety of information is exchanged between these engineers in design and detailing stages. If the coordination between engineers is not performed sufficiently, data exchange may result in loss of important information that may cause inadequate design and detailing of a structure. Thus, a data model developed for different stages of design and detailing of reinforced concrete structures can facilitate the data exchange among engineers and help improving the quality of structural design.

In this study, an object oriented data model was developed for exchanging information for the design and detailing of reinforced concrete columns and beam column joints. The geometry of the structure, amount, shape and placement of reinforcement were defined in this data model. In addition to these, classes that facilitate the design and detailing of reinforced concrete columns and beam column joints according to building codes were also represented.

Another focus of this study is to develop a web based, platform independent data management and multi-user framework for structural design and detailing of reinforced concrete frames. The framework allows simultaneous design of a structure by multiple engineers. XML Web Services technology was utilized for the web based environment in such a way that the design related data was stored and managed centrally by the server in XML files. As a final step, CAD drawings of column reinforcement details in DXF format are prepared.

Keywords: multi-user, web based system, reinforced concrete frames, XML web services, object oriented data mode, column design and detailing

ÖZ

BETONARME ÇERÇEVELER İÇİN AĞ TABANLI TASARIM VE DETAYLANDIRMA ORTAMI

Ünal, Gökhan

Yüksek Lisans, İnşaat Mühendisliği Bölümü

Tez Yöneticisi: Assist. Prof. Dr. Özgür Kurç

Aralık 2009, 93 sayfa

Betonarme yapıların tasarım ve detaylandırmasına birden fazla mühendis katkı sağlamaktadır.Tasarım ve detaylandırma aşamalarında proje çalışanları arasında çok çeşitli veri paylaşımı olmaktadır. Eğer çalışanlar arasındaki koordinasyon yeteri kadar sağlanamaz ise, paylaşılan veride kayıplar olabilir. Bu kayıplar yetersiz ve yanlış tasarıma veya detaylandırmaya, dolayısı ile zaman ve para kaybına neden olabilir. Yaratılacak olan veri modeli, veri kaybını ortadan kaldırarak, ortaya çıkan veriyi yönlendirmeye yardımcı olarak tasarım kalitesini artırır.

Bu çalışmada betonarme kolonlar ve kolon-kiriş birleşim noktaları için nesne tabanlı bir veri modeli geliştirilmiştir. Yapının geometrisi, kullanılacak olan donatının miktarı, şekli ve yerleşimi oluşturulan veri yapısının içinde tanımlanmıştır. Buna ek olarak, kolon ve kolon-kiriş birleşim noktalarının tasarımının ve detaylandırılmasının kolay bir şekilde yönetmeliklere uygun gerçekleştirilmesini sağlamak amacıyla yeni nesneler de geliştirilmiştir.

Bu çalışmanın bir diğer amacı da, birden fazla mühendisin aynı anda betonarme bina üzerinde çalışmasını sağlayacak ağ tabanlı bir ortamın oluşturulmasıdır. Veri, merkezi bir sunucuda bulunmaktadır ve bu sunucunun programlanmasında XML ağ servisleri kullanılmıştır. Veri, sunucuda XML biçiminde tutulmaktadır.

Son olarak, tasarlanan ve detaylandırılan kolonların üç boyutlu teknik çizimleri DXF biçiminde hazırlanmaktadır.

Anahtar Kelimeler: çoklu kullanıcı, ağ tabanlı sistem, betonarme çerçeveler, XML ağ servisleri, nesne tabanlı veri yapısı, betonarme kolon tasarımı ve detaylandırılması

ACKNOWLEDGMENTS

This study was conducted under the supervision of Assist. Prof. Dr. Ozgur Kurc. I would like to express my sincere appreciation for the support, guidance, and insights he has provided me throughout the thesis.

The scholarship provided by The Scientific & Technological Research Council of Turkey (TÜBİTAK) during my graduate study is highly acknowledged

To my family

TABLE OF CONTENTS

ABSTRACT iv
ÖZ vi
ACKNOWLEDGMENTS viii
TABLE OF CONTENTS x
LIST OF FIGURES xiii
LIST OF TABLES xvii
LIST OF SYMBOLS xviii
CHAPTERS

1 INTRODUCTION 1
1.1 Problem Definition 1
1.2 Literature Survey 6
1.2.1 Data Models 6
1.2.2 Web Services 8
1.2.3 Extensible Markup Language (XML) 8
1.3 Objectives and Scope 9
1.4 Thesis Outline 10

2 DESIGN CODES 11
2.1 Introduction 11
2.2 TS500-2000 11
2.2.1 Minimum Requirements for Columns in TS500-2000 11
2.2.2 Moment Magnification Method 12
2.2.2.1 Effective Length Calculation 13
2.2.2.2 Buckling Load 13
2.2.2.3 Moment Magnification Factor 14

2.2.2.4 Conditions in which Second Order Moments can be Neglected . 14
2.2.3 BASIC DEVELOPMENT LENGTH 15
2.3 TURKISH SEISMIC CODE 15
2.3.1 Minimum Requirements for Columns in Turkish Seismic Code 15
2.3.2 Requirements of Having Columns Stronger Than Beams 18
2.3.3 Shear Safety of Columns 19
2.3.4 Beam-Column Joints 20
2.3.4.1 Minimum Transverse Reinforcement in Beam-column Joints . . 22

3 SECTION ANALYSIS UNDER AXIAL LOAD AND BIAXIAL BENDING MOMENT 24
3.1 Introduction 24
3.2 Sections Under Axial Load and Uniaxial Bending Moment 24
3.3 Sections Under Axial Load and Biaxial Bending Moment 27
3.4 Determination of Points on Interaction Surface (Failure Surface) 29
3.5 An Example Problem 32

4 DATA STRUCTURE 36
4.1 Introduction 36
4.2 General Structure 36
4.2.1 Analysis Package 37
4.2.1.1 FrameElement Class 37
4.2.1.2 FrameCrossSection Class 39
4.2.1.3 Node Class 39
4.2.1.4 Station Class 41
4.2.1.5 AnalysisResult Class 41
4.2.1.6 LoadCase Class 42
4.2.1.7 LoadCombination Class 42
4.2.2 Design Entities Package 43
4.2.2.1 Column Class 44
4.2.2.2 Beam Class and BeamStrip Class 46
4.2.2.3 Joint Class 46
4.2.2.4 ColumnLine Class 47
4.2.2.5 Story Class 48
4.2.2.6 Building Class 49

4.2.2.7 DesignSection Class 50
4.2.2.8 MaterialModel Class 51
4.2.3 Reinforcement Package 52
4.2.3.1 ReinforcementPattern Class 53
4.2.3.2 LongitudinalBar Class 54
4.2.3.3 EndStyle Class 56
4.2.3.4 LongitudinalPattern Class 60
4.2.3.5 Pre-defined Longitudinal Reinforcement Patterns 60
4.2.3.6 TransverseBar Class 61
4.2.3.7 TransversePattern Class 62
4.2.3.8 Pre-defined Transverse Reinforcement Patterns 62
4.2.4 Design Code Class 63

5 CASE STUDY 73
5.1 INTRODUCTION 73
5.2 CLIENT GUI 73
5.3 IMPLEMENTATION 76
5.3.1 Initialization Step 79
5.3.2 Simultaneous Design of ColumnLines 80

6 CONCLUSION 86
6.1 FUTURE RECOMMENDATIONS 89

APPENDICES

A CLASS RELATIONSHIP OF DATA MODEL 92

B COORDINATE TRANSFORMATION METHOD 93

LIST OF FIGURES

FIGURES

Figure 1.1 Structural Design Process of Reinforced Concrete Structures 2
Figure 1.2 Participants in a single project (International Alliance for Interoperability) 3
Figure 1.3 The normal method of data exchange for the case in Figure 1.2 (International Alliance for Interoperability) 5
Figure 1.4 (a) The normal method of data exchange ,(b) An approach using a neutral file (Eastman, 1999) 5
Figure 1.5 Reinforcement Bar definition in IFC (IAI, 2009) 7
Figure 1.6 Sample XML Code 9

Figure 2.1 Longitudinal reinforcement in beam-column joints (TEC-2007) 16
Figure 2.2 Unfavorable Earthquake Direction (TEC-2007) 19
Figure 2.3 M_a and $M_{\ddot{u}}$ in column shear design 20
Figure 2.4 Shear force in beam-column joint (TEC-2007) 21
Figure 2.5 Shear Reinforcement Detailing in Columns (TEC-2007) 23

Figure 3.1 A Section Under Axial Load and Uniaxial Bending Moment 25
Figure 3.2 Typical Moment Interaction Diagram 27
Figure 3.3 A Section Under Axial Load and Biaxial Bending Moment 27
Figure 3.4 Typical Biaxial Interaction Diagram 28
Figure 3.5 Pseudo Code for Calculation of Points of Failure Surface 31
Figure 3.6 An example Problem 32
Figure 3.7 Analysis of section rotated by 0^o 32
Figure 3.8 Analysis of section rotated by 30^o 33
Figure 3.9 Analysis of section rotated by 60^o 34

Figure 3.10 Analysis of section rotated by 90^o 34
Figure 3.11 M33 - M22 diagram for example section 35

Figure 4.1 General architecture of the object library 37
Figure 4.2 Analysis Package 38
Figure 4.3 Major Elements In a Structure 38
Figure 4.4 Attributes in FrameElement Class 38
Figure 4.5 Attributes in FrameCrossSection Class 39
Figure 4.6 Frame elements connecting to a single node 40
Figure 4.7 Attributes in Node Class 40
Figure 4.8 Attributes in Point Class 40
Figure 4.9 Attributes in Displacement Class 41
Figure 4.10 Attributes in Station Class 41
Figure 4.11 Attributes in AnalysisResult Class 42
Figure 4.12 Attributes in LoadCase Class 42
Figure 4.13 Class Relationship of Building Class 43
Figure 4.14 Attributes in LoadCombination Class 43
Figure 4.15 Attributes in CombinationParams Class 43
Figure 4.16 Design Entities Package 44
Figure 4.17 Columns may be entered as more than one piece 45
Figure 4.18 Attributes in Column Class 45
Figure 4.19 Class Relationship of Column Class 46
Figure 4.20 Class Relationship of Joint Class 47
Figure 4.21 Examples of Column Line 47
Figure 4.22 Attributes in ColumnLine Class 48
Figure 4.23 Class Relationship of ColumnLine Class 48
Figure 4.24 Attributes in Story Class 49
Figure 4.25 Class Relationship of Story Class 49
Figure 4.26 Class Relationship of Building Class 50
Figure 4.27 Attributes in DesignSection Class 50
Figure 4.28 Class Relationship of MaterialModel Class 52
Figure 4.29 Reinforcement Package 53
Figure 4.30 Attributes in ReinforcementPattern Class 54
Figure 4.31 Attributes in Rebar Class 54

Figure 4.32 Basic Straight Part of a LongitudinalBar and Possible Alternatives . . 55
Figure 4.33 Attributes in LongitudinalBar Class 56
Figure 4.34 Splice is done at the floor level 56
Figure 4.35 Splice is done at the mid-height of the column 57
Figure 4.36 Second type of the end-style shape 57
Figure 4.37 Third type of the end-style shape 58
Figure 4.38 Splice is done at the floor level 59
Figure 4.39 Splice is done at mid-height of the column 59
Figure 4.40 Different Directions of Longitudinal Reinforcement at Beam-Column Joints 59
Figure 4.41 Attributes in EndStyle Class 60
Figure 4.42 Attributes in LongitudinalPattern Class 60
Figure 4.43 Pre-defined Longitudinal Reinforcement Patterns 61
Figure 4.44 Attributes in TransverseBar Class 62
Figure 4.45 Attributes in TransversePattern Class 62
Figure 4.46 Pre-defined Transverse Reinforcement Patterns 63
Figure 4.47 Overwritten Methods in Code Class 64
Figure 4.48 Class Relationship of Code Class 65
Figure 4.49 Methods in TS500_2000 and TEC 66

Figure 5.1 General view of the client GUI 74
Figure 5.2 Concrete Tab of Project Preferences 75
Figure 5.3 Reinforcement Tab of Project Preferences 75
Figure 5.4 Preferences Tab of Project Preferences 76
Figure 5.5 Design Tab of Project Preferences 76
Figure 5.6 3D view of the sample project 77
Figure 5.7 Flow Chart to Design a ColumnLine 78
Figure 5.8 Pseudo Code for Detection of Frame Elements in the Same Column . . 80
Figure 5.9 View of the sample project at client GUI 81
Figure 5.10 Selected ColumnLine in the sample project 82
Figure 5.11 Selection of predefined patterns at client GUI 83
Figure 5.12 Section view of selected column line 84
Figure 5.13 Side view of selected column line 84
Figure 5.14 3D view of selected column line 85

Figure A.1 Class Relationship of Data Model 92

Figure B.1 Rotation of coordinate system 93

LIST OF TABLES

TABLES

Table 4.1 Surface texture enumerator 54

LIST OF SYMBOLS

A_c	Cross-sectional area of the concrete section.
A_{st}	Longitudinal reinforcement area.
E_c	Modulus of elasticity of concrete.
EI	Effective flexural rigidity.
i	Radius of gyration.
i_{33}	Radius of gyration of a section about its major axis.
i_{22}	Radius of gyration of a section about its minor axis.
I_c	Moment of inertia of gross concrete section.
I_{33}	Moment of inertia of a section about its major axis.
I_{22}	Moment of inertia of a section about its minor axis.
k	A multiplier that depends on whether the frame is sway or not and is a function of relative stiffness of columns and beams at the beam-column joints above and below.
l_i	Length of each column in ith floor. Length is measured form the center of the joint to sthe center of the joint.
l_k	The effective length of column.
l_n	Clear height of the column. It is measured from the top of the slab to the bottom of the beam above.
M_{d1}, M_{d2}	Design end moments found from analysis.
N_{di}	Axial force in each column.
N_{gd}	Design sustained axial load.
N_d	Total design axial load.
V_{gd}	Sum of the design shear forces in the floor caused by the sustained load.
V_d	Sum of the design shear force.
V_{fi}	Sum of horizontal shear forces in ith floor.
R_m	Factor that reflects the effect of creep.
S_{33}	Section modulus of a section about its major axis.
S_{22}	Section modulus of a section about its minor axis.
ρ_t	Longitudinal reinforcement area.
Δ_i	Lateral displacement of ith floor, relative to the floor below.

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

In order to design and detail reinforced concrete structures, there are many steps to follow : "Preliminary Design", "Modeling and Analysis", "Design and Detailing of Reinforcement", "Code Checking" and "Structural Drawings". The interaction of these steps are presented in Figure 1.1. The first step in designing and detailing of a reinforced concrete structure is the "Preliminary Design" step. In this step, structural framing, initial member sizes, material properties, support conditions, ductility level of the structure and loads to be applied to the structure are determined. As a next step, the reinforced concrete structure is modeled and analyzed in an analysis program with the properties defined in the "Preliminary Design" step". Element forces and displacements are obtained at the end of this step. In "Design and Detailing of Reinforcement" step, amount, location and shape of the reinforcement are determined by using the current member sizes and the element forces obtained in the previous step. In details, the diameter of longitudinal reinforcement, longitudinal reinforcement mesh in the column or beam cross-section, the shape of the longitudinal reinforcement at beam-column joints, length of the longitudinal reinforcement according to splice location, the diameter of transverse reinforcement, spacing of transverse reinforcement at confined and unconfined regions of columns and beams, transverse reinforcement detailing at beam-column joints are determined in this step. Then, the structure is checked according to a specified design code in "Code Checking" step. Minimum cross-sectional dimensions and area, minimum and maximum longitudinal reinforcement amount and diameter, minimum transverse reinforcement diameter, maximum transverse reinforcement spacing, maximum deflection are some of the criteria that a design code checks. At any step of this design process, it may be realized that the requirements of the design codes cannot be satisfied with the current

design decisions. In such cases, the structural framing or the current member dimensions are revised and the design process is repeated by reanalyzing the structural models of the building. This indicates that design and detailing of reinforced concrete structures are not serial processes but iterative ones between the analysis and design steps. Finally, structural drawings that indicate the member dimensions and placement of reinforcement are prepared in "Structural Drawings" step.

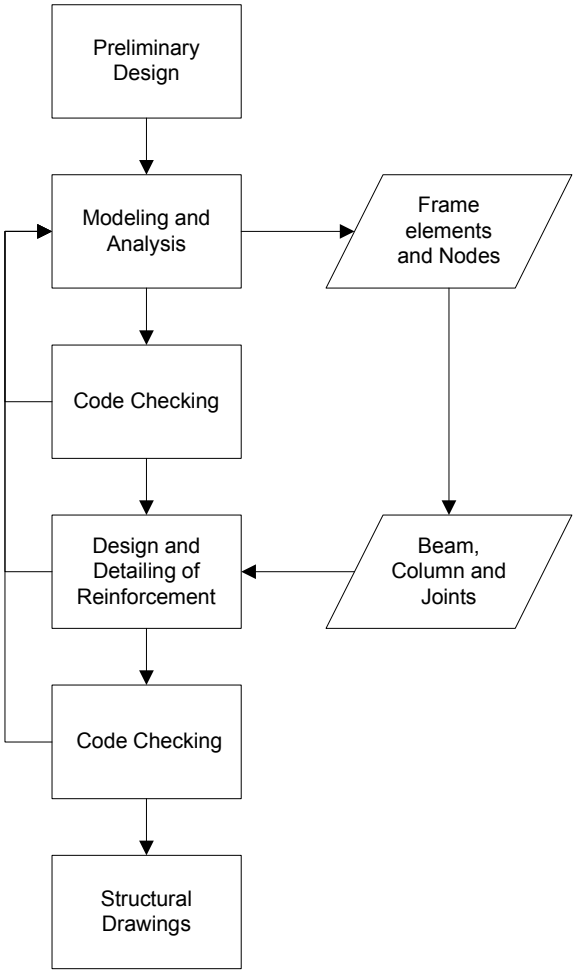

Figure 1.1: Structural Design Process of Reinforced Concrete Structures

Analysis and Design steps require different kinds of information. In the "Modeling and Analysis" step, one dimensional frame elements and nodes are used. These frame elements, however, are not enough to design and detail the reinforced concrete structures. Therefore,

higher level geometrical objects such as beam, column and reinforcement are required in "Design and Detailing of Reinforcement" step. Because of this reason, structural design process of reinforced concrete structures includes complex information exchanges between the steps defined in Figure 1.1.

Due to the the monolithic nature of cast-in-place reinforced concrete members, the design of beams and columns is performed system based. In other words, continuous beams are often designed as a whole. Similarly, columns on the same grid points are designed and detailed continuously from the foundation level to the top of the building. Moreover, the reinforcement detail of a member affects the way the neighbor members are designed and detailed. Especially, in seismic design, the flexural capacities of beams of the special moment frames are the key elements that determine the amount of shear force that can be transferred to a column or a beam-column joint.

In engineering projects, people from many different professions contribute to a single project such as architects, technical drawers, structural engineers, soil engineers, mechanical engineers, project managers, controllers, etc. [Figure 1.2]. They have all specific and different opinions about the same project before, during and even after the project. These people come together and share their opinions in the progress of the project. Sometimes, they have meetings, make conversations, send e-mails to each other, post drawings or reports, etc. to be synchronic. All these procedures, however, may not be enough to be synchronic due to the reasons described in the following paragraphs.

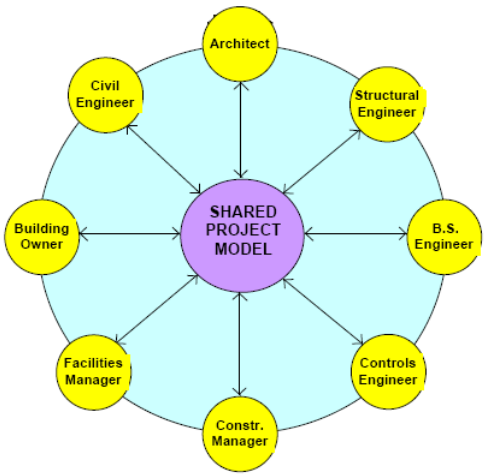

Figure 1.2: Participants in a single project (International Alliance for Interoperability)

People involved in a project may use different software programs related to their profession

and format of data reports generated from these programs may be different from each other. To use data reports in different formats means that either the program to be used should be able to run for different formats or data reports should be translated to required format manually. Since integration of software systems is still a developing research area, the second method is used which means that data loss is inevitable. Even if the same data format is used while exchanging data, there are many reasons to lose data. Information not managed centrally, use of different versions of software programs, unreceived e-mails or posts, unsynchorinized information and human factors are some of the reasons that cause loss of data and time.

Besides data and time loss, data incompatibility causes huge amount of money loss. The National Institute for Standards and Technology imposes at least one billion dollar per year on the members of the U.S. automotive supply chain and has estimated that data incompatibility is a 90 billion dollar problem for manufacturing industry [Loffredo]. Moreover, the estimated cost of data incompatibility to the U.S construction industry was 15.8 billion dollars in 2002 [Gallaher].

The most important reason that cause data loss is the information that is not managed centrally. For example, if there are two project participants: A and B, and data is exchanged in both direction between them, there should be two translators considering normal method : A-to-B and B-to-A. If there are three project participants, however, there should be six translators : A-to-B, B-to-A, A-to-C, C-to-A, B-to-C and C-to-B. When the case in Figure 1.2 is considered, the number of translators should be 56 (Figure 1.3). Generally, the number of required translators between N project participants is $N * (N - 1)$,Figure 1.4(a). On the other hand, if centrally managed information model is used while exchanging data, the number of translators between participants reduces to $2 * N$ Figure 1.4(b) which shows the importance of using information model.

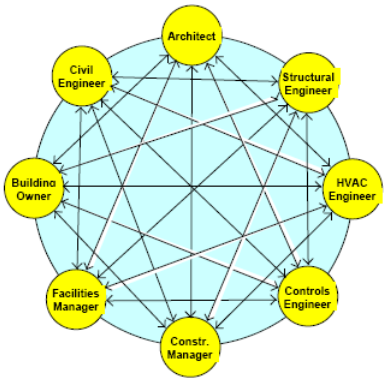

Figure 1.3: The normal method of data exchange for the case in Figure 1.2 (International Alliance for Interoperability)

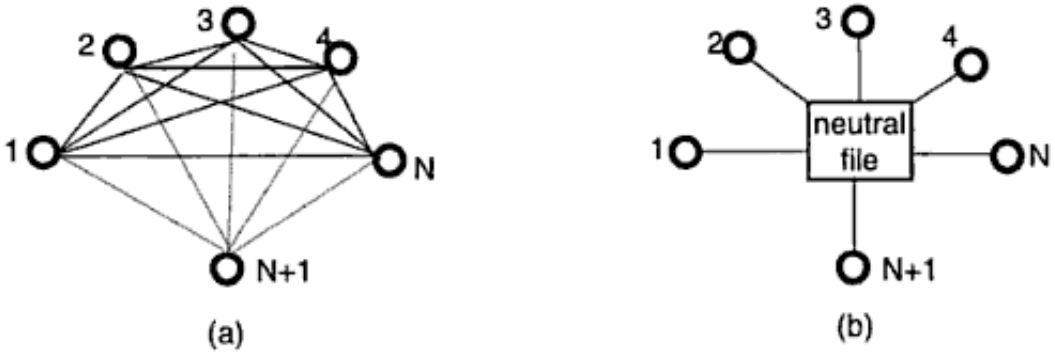

Figure 1.4: (a) The normal method of data exchange ,(b) An approach using a neutral file (Eastman, 1999)

It is possible to encounter with the same problem in design and detailing of reinforced concrete structures. There may be more than one engineer and draftsmen in design and detailing of reinforced concrete frames. Different members of structure can be designed and detailed by different engineers and different draftsmen can prepare the structural drawings of detailed members. When there is a failure to update or missing information between engineers and draftsmen working in the same project, this may cause incorrect design and detailing of reinforced concrete members which means inadequacy in structural safety, time and money loss. Thus, an environment that keeps neutral data files and allows multiple engineers work on a single project improve the quality of design and detailing of reinforced concrete structures. In a such a neutral data model, all entities related reinforced concrete members and reinforcement should be defined. Moreover, code based calculations should be

performed with this data model.

Therefore, the aim of this study is to develop an object oriented library to facilitate the design and detailing of reinforced concrete frames. Moreover, to develop an environment in which more than one engineer perform code based design and detailing tasks simultaneously on a single reinforced concrete project with this developed library is another aim of this study.

1.2 Literature Survey

1.2.1 Data Models

The existence of incompatibilities between information technology systems used in the processes of design, engineering, and manufacturing has resulted in the development of several different approaches for achieving the exchange of data. One of these approach is STEP (Standard for the Exchange of Product modal data). STEP was launched in 1984 by ISO (International Standardization Organization) under the umbrella of ISO 10303. The fundamental aim of STEP is to exchange data and share information. STEP addresses product data from mechanical and electrical design, geometric dimensioning and tolerancing, analysis and manufacturing, with additional information specific to various industries such as automotive, aerospace, building construction, ship, oil and gas, process plants and others. STEP uses EXPRESS which is a standard data modeling language for product data and is standardized as IS0 10303-11 by ISO. Application data in STEP is exchanged either by a *STEP-File*, most widely used exchange form of STEP and whose format is defined in ISO 10303-21, *STEP-XML*, an alternative way to STEP-File and specifies the use of the Extensible Markup Language (XML) to represent EXPRESS.

According to Fowler, at the initial stages of STEP, building and construction were not very active branches within STEP. With the following three application protocols, AP225: "Building elements using explicit shape representation", AP228: "Building services: heating, ventilation and air conditioning" and AP230: "Building structural frame: steel work", the role of STEP in these branches became widespread and within the collaborative projects such as ATLAS (ESPRIT program), COMBINE and COMBINE 2 (JOULE program) and CIMSTEEL (EUREKA program) these application protocols have been developed.

Another approach to exchange data without incompatibilities is IFC (Industry Foundation Classes). IFC was developed by International Alliance for Interoperability (IAI) in 1995.

Although, it has still not been fully accepted, it is the most utilized platform for interoperability [Ilal]. More than 600 organizations around the world are committed to producing and using IFC standard object definitions [Coble]. The main goal of IFC is to facilitate interoperability in the building industry. The IFC data model is an object-oriented data model based on class definitions representing the things (elements, processes, shapes, etc.) that are used by software applications during a construction or facility management project. The IFC data model focuses on those classes that are needed to share information (rather then processing it in a particular software). The IFC data model is a neutral and open specification that is not controlled by a singular vendor or group of vendors. The Japanese Chapter of IAI incorporated the reinforced concrete structural model into IFC as an extension. In this extension, reinforcing bars were defined with the "IfcReinforcingBar Class" with the properties of diameter, length, cross-sectional area, usage of reinforcement, and surface texture (Figure 1.5). Although this extension covered definitions of geometry of reinforcing bars in reinforced concrete members, design code based calculations and detailing of reinforcement at special zones were not handled in IFC.

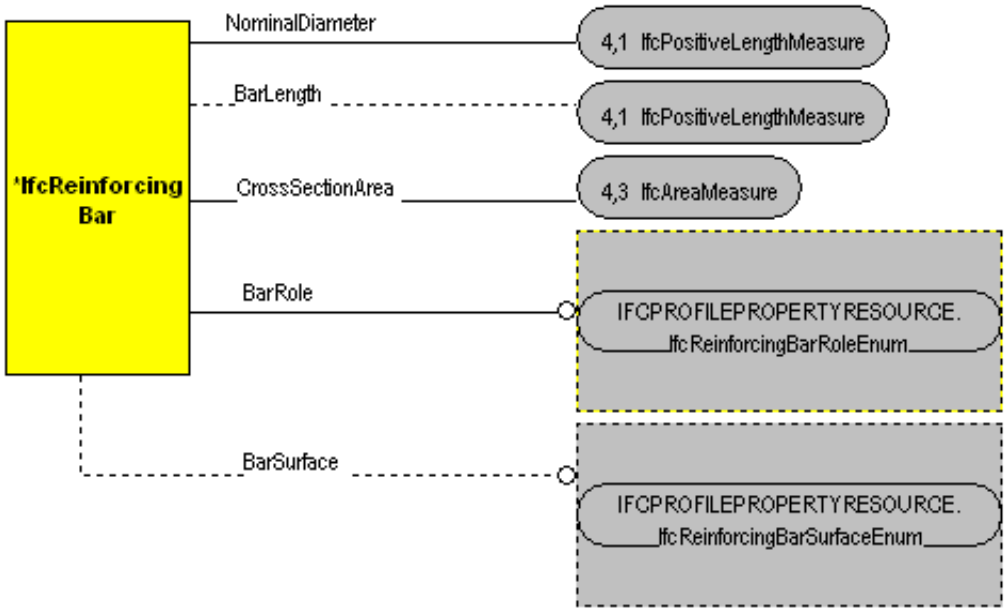

Figure 1.5: Reinforcement Bar definition in IFC (IAI, 2009)

1.2.2 Web Services

A Web Service is defined by the World Wide Web Consortium (W3C) as a software system designed to support interoperable machine-to-machine interaction over a network. It uses XML, SOAP, WSDL and UDDI open standards. XML is used in Web Services so that information can be exchanged between disparate application and platforms. SOAP, Simple Object Access Protocol, is a messaging protocol for transporting information and instructions between Web Services. Web Services Description Language (WSDL) provides a standard method to describe Web Services and Universal Description, Discovery and Integration (UDDI) defines standard rules for Web Service directories and is used for listing what services are available. Both WSDL and UDDI are based on XML.

The most important advantage of Web Services over previously distributed-computing technologies (DCOM, CORBA) is that Web Services use open standards (XML, SOAP, WSDL, UDDI) [Deitel, 2003]. It means that Web Services enable any two software components whose programming languages and platforms are different to communicate. In other words, Web Services are platform-independent.

1.2.3 Extensible Markup Language (XML)

Extensible Markup Language (XML) is a way of representing information hierarchically and lossless exchange of complex data between systems that use different formats can be achieved with XML. It was developed by World Wide Web Consortium (W3C) for a new version of Standard Generalized Markup Language (SGML) in 1996. In a few years, XML was used as the basis of many technologies. These applications of XML helped make the technology less abstract and more real [William, 2002]. Then, which properties of XML made it so powerful? First of all, XML uses human, not computer, language. Then, it is readable by even people who have no formal introduction on XML. Then, XML parsers are quickly available. Finally, XML is extendable. On the other hand, the major disadvantage of XML is that XML files can become very large easily. Size of the XML node of a value is always larger than the size of the stored value itself due to start and end tag of the value.

XML documents have an object oriented structures composed of blocks of data. Every data block in an XML document is enclosed between tags (start tag and end tag) and is called as a node. Start tag starts with "<" and ends with ">", end tag starts with "</" and ends with ">". Nodes can contain other child nodes. As it can be seen in figure Figure 1.6, *ColumnForces* is the main node in this sample XML code. *ColumnForces* node includes a

Story node with an identifier attribute "story" whose value is "STORY1". Moreover, *Story* node contains two *Column* nodes with an identifier attribute "ID" whose values are "C4" and "C18". In each *Column* node, there are two *LoadCase* node with an identifier attribute "load", namely "DEAD" and "LIVE". Each *LoadCase* node has 6 properties, namely "P", "V2", "V3", "T", "M2" and "M3". Before ending a main node, child nodes in the main node should be closed. For example, in figure Figure 1.6, to end *Story* node with </Story>, *Column* node should be ended with </Column>.

```
<?xml version="1.0" encoding=""utf-8" ?>
-<ColumnForces>
 -<Story story="STORY1">
  -<Column ID="C4">
   -<LoadCase load="DEAD">
     <P>-10000</P>
     <V2>-15000</V2>
     <V3>-10000</V3>
     <T>0</T>
     <M2>-10000</M2>
     <M3>-10000</M3>
   </LoadCase>
  -<LoadCase load="LIVE">
    <P>10000</P>
    <V2>20000</V2>
    <V3>0</V3>
    <T>0</T>
    <M2>0</M2>
    <M3>35000</M3>
   </LoadCase>
  </Column>
 </Story>
</ColumnForces>
```

Figure 1.6: Sample XML Code

1.3 Objectives and Scope

The main objective of this study is to develop an object oriented library to facilitate the design and detailing of reinforced concrete columns and beam-column connection zones by multiple engineers simultaneously. The object library involves object abstractions, definitions of relations between objects and implementation of object functionality for performing all required design tasks related to the design process of reinforced concrete columns. Implementation of the developed libraries into a web based system, which allows multiple engineers work on a single project simultaneously, is the second objective of this study. Final objective of this study is to test the performance and functionality of developed object oriented library on an actual model.

In this research, the web-based platform that was previously developed by Anıl [Anıl, 2009]

will be extended. The new extensions are summarized as follows :

- Code based design and detailing of reinforced concrete columns amd beam-column joints.
- A simple client GUI (Graphical User Interface) is developed to visualize RC building and select RC columns to be designed and detailed.
- Detailed drawings, which are prepared at the end of the design and detailing of reinforced concrete columns and beam-column joints, are in AutoCAD dxf format. These prepared drawings are not used as detailed shop-drawings but they are used to visualize reinforcement arrangement in columns in 3D views.

For this purpose, C# .NET programming language is utilized for the development of the object oriented data model and XML (eXtensible Markup Language) Web Services are used to build web based multi-user environment.

1.4 Thesis Outline

The remainder of thesis is organized as follows: Chapter 2 presents requirements and limitations for design and detailing of reinforced concrete columns according TS500-2000 and Turkish Earthquake Code. In Chapter 3, analysis procedure of reinforced concrete sections under axial load and biaxial bending moment is presented. Then, drawing procedure for biaixal interaction diagram for specified axial load is presented. Chapter 4 is devoted to elements of data model. In this chapter, analysis and design layer are described in detail and reinforcement and code classes are introduced. In Chapter 5, a case study for an actual model is introduced and results are discussed. Finally, Chapter 6, is the final chapter which summarizes the efficiency and future plans of this study.

CHAPTER 2

DESIGN CODES

2.1 Introduction

In this study, two codes are used for designing and detailing of reinforced concrete columns and beam-column joints, namely Turkish Reinforced Concrete Design Code (TS500-2000) and Turkish Earthquake Code (TEC-2007). These codes are used to check minimum column dimensions, minimum and maximum diameter, spacing and reinforcement amount for longitudinal and transverse reinforcement. Moreover, beam-column joints are also checked for design shear force and detailing of shear reinforcement at these regions.

2.2 TS500-2000

2.2.1 Minimum Requirements for Columns in TS500-2000

According to TS00-2000, cross-sectional requirements, longitudinal reinforcement requirements, transverse reinforcement requirements are stated as follows:

- Cross-sectional Requirements:
  - Minimum cross-sectional dimension :
    - 250 mm for rectangular columns.
    - 300 mm in diameter for circular columns.
  - Minimum cross-sectional area:
    - $A_c \geq \frac{N_d}{0.6 f_{ck}} \geq 75000\, mm^2$
- Longitudinal Reinforcement Requirements:

– Minimum longitudinal reinforcement ratio:

$$\rho_t \geq 0.01 \tag{2.1}$$

– Maximum longitudinal reinforcement ratio:

$$\rho_t \leq 0.04 \tag{2.2}$$

– Minimum diameter of longitudinal bar:

$$\geq \phi 14 \tag{2.3}$$

- Transverse Reinforcement Requirements:

  – The diameter of the transverse reinforcement should be greater than the 1/3 of the maximum diameter of longitudinal bars and 8 mm.

$$\phi_{transverse} \geq 1/3\, max\phi_{longitudinal} \tag{2.4}$$

$$\phi_{transverse} \geq 8\, mm \tag{2.5}$$

  – The spacing between the transverse reinforcement should be smaller than 12 times minimum diameter of longitudinal bars and 200mm.

$$s_{transverse} \leq 12\, min\phi_{longitudinal} \tag{2.6}$$

$$s_{transverse} \leq 200\, mm \tag{2.7}$$

2.2.2 Moment Magnification Method

When buildings are subjected to lateral loads, relative displacement between columns ends exists. This relative displacement increases initial eccentricity of the column and therefore causes additional moment in the column. This additional moment is called as "Second Order Moment". Second order moment depends on the geometry of the deflected shape and slenderness of the column.

In order to take the slenderness effects while calculating the design moment of a column, "Moment Magnification Method" is introduced in TS500-2000. In this method, first order design moment is multiplied by a factor β. This factor is calculated differently for braced and unbraced frames. Therefore, as a first step it should be checked whether the frame is braced against side-sway or not.

According to TS500-2000, to check whether a story is braced (non-sway) or not, it is stated that if the parameter ψ defined in Equation 2.8 is equal to less than 0.05, the story can be considered as braced (non-sway); otherwise, it is unbraced (sway).

$$\psi = 1.5\Delta_i \frac{\sum \frac{N_{di}}{l_i}}{V_{fi}} \leq 0.05 \tag{2.8}$$

In Δ_i,N_{di} and V_{fi} calculations, only lateral load combinations (earthquake or wind) should be considered.

2.2.2.1 Effective Length Calculation

The effective length of a column is calculated by multiplying clear height of the column by a factor "k" which takes into account the boundary conditions of column. The multiplier "k" depends on whether the frame is sway or not and is a function of relative stiffness of columns and beams at the beam-column joints above and below. The multiplier "k" is calculated as follows:

- For Non-sway Frames:

$$k = 0.7 + 0.05(\alpha_1 + \alpha_2) \leq (0.85 + 0.05\alpha_1) \leq 1.0 \tag{2.9}$$

- For Sway Frames:

$$k = \frac{20 - \alpha_m}{20}\sqrt{1 + \alpha_m} \text{ if } \alpha_m < 2 \tag{2.10}$$

$$k = 0.9\sqrt{1 + \alpha_m} \text{ if } \alpha_m \geq 2 \tag{2.11}$$

  where $\alpha_{1,2} = \dfrac{\sum (I/l)_{column}}{\sum (I/l)_{beam}}$ and $\alpha_m = 0.5(\alpha_1 + \alpha_2)$. In calculating α values (relative stiffness values), moment of inertia of beams should be based on cracked section and only beams in bending direction should be taken into consideration.

- For Columns with Hinge at One End:

$$k = 2.0 + 0.3\alpha \tag{2.12}$$

  where α is the value at the joint where the column is no hinged.

2.2.2.2 Buckling Load

According to TS500-2000, column buckling load N_k, is calculated as follows:

$$N_k = \frac{\pi^2 EI}{l_k^2} \tag{2.13}$$

where

$$EI = \frac{E_c I_c}{2.5} \frac{1}{1 + R_m} \tag{2.14}$$

$$R_m = \frac{N_{gd}}{N_d} \text{ for Braced Frames} \tag{2.15}$$

$$R_m = \frac{\sum V_{gd}}{\sum V_d} \text{ for Unbraced Frames} \tag{2.16}$$

2.2.2.3 Moment Magnification Factor

According to TS500-2000, moment magnification factor for non-sway and sway frames is calculated as follows:

- For Non-Sway Frames:

$$\beta = \frac{C_m}{1 - 1.3\frac{N_d}{N_k}} \tag{2.17}$$

  where $C_m = 0.6 + 0.4(M_{d1}/M_{d2}) \geq 0.4$ and $M_{d1} \leq M_{d2}$. If the column is bent in single curvature, M_{d1}/M_{d2} is taken as positive. If it is bent in double curvature, M_{d1}/M_{d2} is taken as negative.

- For Sway Frames:

$$\beta_s = \frac{1}{1 - 1.3\frac{\sum N_d}{\sum N_k}} \geq 1.0 \tag{2.18}$$

  In TS500-2000, it is stated that β values should be calculated for all columns in an unbraced system by using Equation 2.18 taking $C_m = 1.0$. In designing the column, the maximum of β and β_s should be used.

The design moment including the second order effects is calculated by multiplying the first order design moment by β.

2.2.2.4 Conditions in which Second Order Moments can be Neglected

In TS500-2000 it is stated the second order moments can be neglected if the following equations are satisfied. In such a case, the maximum moment obtained from the first order analysis is taken as the design moment of the column.

- For Braced Frames:

$$\frac{l_k}{i} \leq 34 - 12\frac{M_{d1}}{M_{d2}} \leq 40 \tag{2.19}$$

- For Unbraced Frames:

$$\frac{l_k}{i} \leq 22 \tag{2.20}$$

If the column is bent in single curvature, M_{d1}/M_{d2} is taken as positive. If it is bent in double curvature,M_{d1}/M_{d2} is taken as negative.

2.2.3 BASIC DEVELOPMENT LENGTH

In TS500-2000, it is stated that the basic development length l_b, is calculated as follows:

- For deformed bars:

$$l_b = 0.12\frac{f_{yd}}{f_{ctd}}\phi \geq 20\phi \tag{2.21}$$

- For plain(undeformed) bars:

$$l_b = 0.24\frac{f_{yd}}{f_{ctd}}\phi \geq 40\phi \tag{2.22}$$

If the diameter of longitudinal bar is $32mm < \phi \leq 40mm$, the development length computed should be multiplied by a factor which is $\frac{100}{132-\phi}$

2.3 TURKISH SEISMIC CODE

2.3.1 Minimum Requirements for Columns in Turkish Seismic Code

In Turkish Seismic Code, cross-sectional requirements, longitudinal reinforcement requirements, transverse reinforcement requirements are stated as follows:

- Cross-sectional Requirements:
  - Minimum cross-sectional dimension :
    - 250 mm for rectangular columns.
    - 300 mm in diameter for circular columns.
  - Minimum cross-sectional area:
    - $A_c \geq \frac{N_d}{0.5f_{ck}} \geq 75000\, mm^2$
- Longitudinal Reinforcement Requirements:
  - Minimum longitudinal reinforcement ratio:

$$\rho_t \geq 0.01 \tag{2.23}$$

– Maximum longitudinal reinforcement ratio:

$$\rho_t \leq 0.04 \tag{2.24}$$

– Minimum diameter of longitudinal bar:

$$\geq \phi 14 \tag{2.25}$$

– Turkish Seismic Code imposes the following requirements for the column longitudinal bars:

  * If lap splices are made at the mid-height of the column, the splice length is equal to the basic development length. $l_0 = l_b$ as calculated according to TS500-2000.

  * If lap splices are made at floor level, the splice length should not be less than the values given below:

    · If only 50% or less of bars is lapped at the same section, the splice length should be greater than 1.25 times basic development length. $l \geq 1.25 l_b$

    · If more than 50% of the bars are lapped at the same section, the splice length should be greater than 1.5 times basic development length. $l_0 \geq 1.5 l_b$

– When column size changes in the floor above, the longitudinal reinforcement of the column should be detailed as shown:

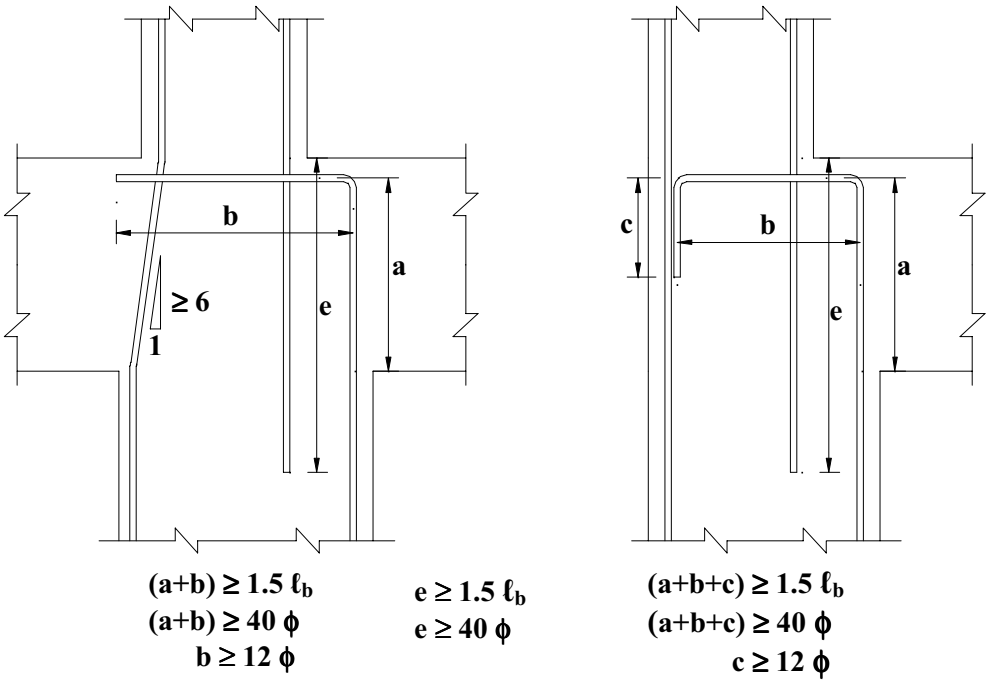

Figure 2.1: Longitudinal reinforcement in beam-column joints (TEC-2007)

- Transverse Reinforcement Requirements

  - According to Turkish Seismic Code, there should be special confinement zones at the bottom and top ends of each column. These zones are called "Column Confinement Zone" and the zone between the confinement zones is called "Column Central Zone" (Figure Figure 2.5).

    * Column Confinement Zone :

      · The length of each column confinement zone shall not be less than maximum of column cross-section dimensions (b_{max}), 1/6 clear length of the column (l_n) and 500 mm.

$$\begin{aligned} &\geq b_{max} \\ &\geq l_n/6 \\ &\geq 500\,mm \end{aligned} \tag{2.26}$$

      · The minimum transverse reinforcement diameter that shall be used in column confinement zones is 8 mm.

$$\phi_{min} \geq 8\,mm \tag{2.27}$$

      · The spacing between the transverse reinforcement in the column confinement zone shall not be less than 50 mm, and shall not be grater than 1/3 the smaller cross-sectional dimension and 100 mm.

$$\begin{aligned} s_c &\geq 50\,mm \\ s_c &\leq 100\,mm \\ s_c &\leq b_{min}/3 \end{aligned} \tag{2.28}$$

      · In case where $N_d > 0.2A_cf_{ck}$, minimum total area of transverse reinforcement to be used in confinement zones shall satisfy both of the following conditions. In this calculation, core diameter of column , shall be considered separately for each direction. (Figure 1131313).

$$A_{sh} \geq 0.3sb_k[(A_c/A_{ck}) - 1](f_{ck}/f_{ywk}) \tag{2.29}$$

$$A_{sh} \geq 0.075sb_k(f_{ck}/f_{ywk}) \tag{2.30}$$

- In the case where $N_d \leq 0.2 A_c f_{ck}$ minimum total area of transverse reinforcement to be used in confinement zones shall be at least 2/3 the transverse reinforcement given by Equation 2.29 and Equation 2.30.

* Column Middle Zone :

  - The minimum transverse reinforcement diameter that shall be used in column central zones is 8 mm.

$$\phi_{min} \geq 8\, mm \tag{2.31}$$

  - The spacing between transverse reinforcement in column central zone shall not be grater than 1/2 the smaller cross-sectional dimension and 200 mm.

$$s_0 \leq 200\, mm \tag{2.32}$$

$$s_0 \leq b_{min}/2 \tag{2.33}$$

2.3.2 Requirements of Having Columns Stronger Than Beams

In a beam-column joint, if the sum of ultimate moment resistances of columns is at least 20% more than sum of the moment resistances of beams, it is considered columns are stronger than beams in this particular beam-column joint (Equation 2.34).

$$M_{ra} + M_{r\ddot{u}} \geq 1.2(M_{ri} + M_{rj}) \tag{2.34}$$

Equation 2.34 shall be applied separately for both earthquake direction and sense to yield the most unfavorable result (Figure 2.2). In calculating column ultimate moment resistance, axial force N_d, shall be taken to yield minimum moments consistent with the sense of earthquake direction.

If $N_d \leq 0.1 A_c f_{ck}$ in both columns at a beam-column joint, there is no need to check whether columns are stronger than beams or not for that particular beam-column joint.

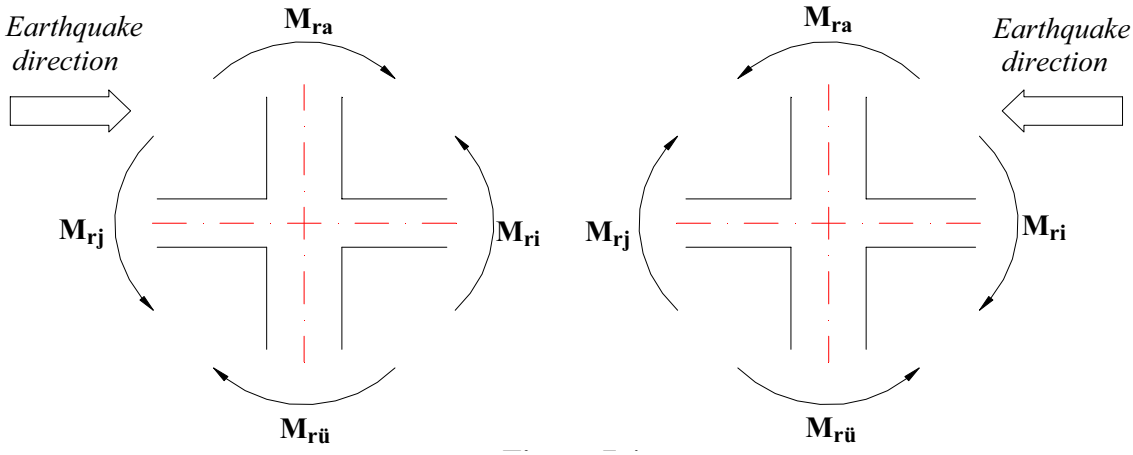

Figure 2.2: Unfavorable Earthquake Direction (TEC-2007)

2.3.3 Shear Safety of Columns

To design column transverse reinforcement, the design shear force V_e, shall be calculated by Equation 2.35

$$V_e = (M_a + M_{\ddot{u}})/l_n \tag{2.35}$$

Design shear force, obtained according to Equation 2.35 shall not be less than the shear force calculated under factored gravity and seismic loads combined. Moreover, the design shear force shall not be greater than the values given by Equation 2.36. If the second condition in Equation 2.36 is not satisfied, the column cross-sectional dimensions shall be increased and seismic analysis shall be repeated according to updated dimensions.

$$\begin{aligned} V_e &\leq V_r \\ V_e &\leq 0.22 A_w f_{cd} \end{aligned} \tag{2.36}$$

In order to obtain M_a and $M_{\ddot{u}}$ in calculation of column design shear force, first of all, top and bottom joints of column shall be checked whether the columns are stronger than beams. If so, the sum of ultimate moment capacities of beams connecting to column in specified direction shall be distributed to columns in proportion to column stiffness. The distributed moments are used in Equation 2.35 as M_a and $M_{\ddot{u}}$. Otherwise, column moment capacities shall be used in Equation 2.35 as M_a and $M_{\ddot{u}}$(Figure 2.3).

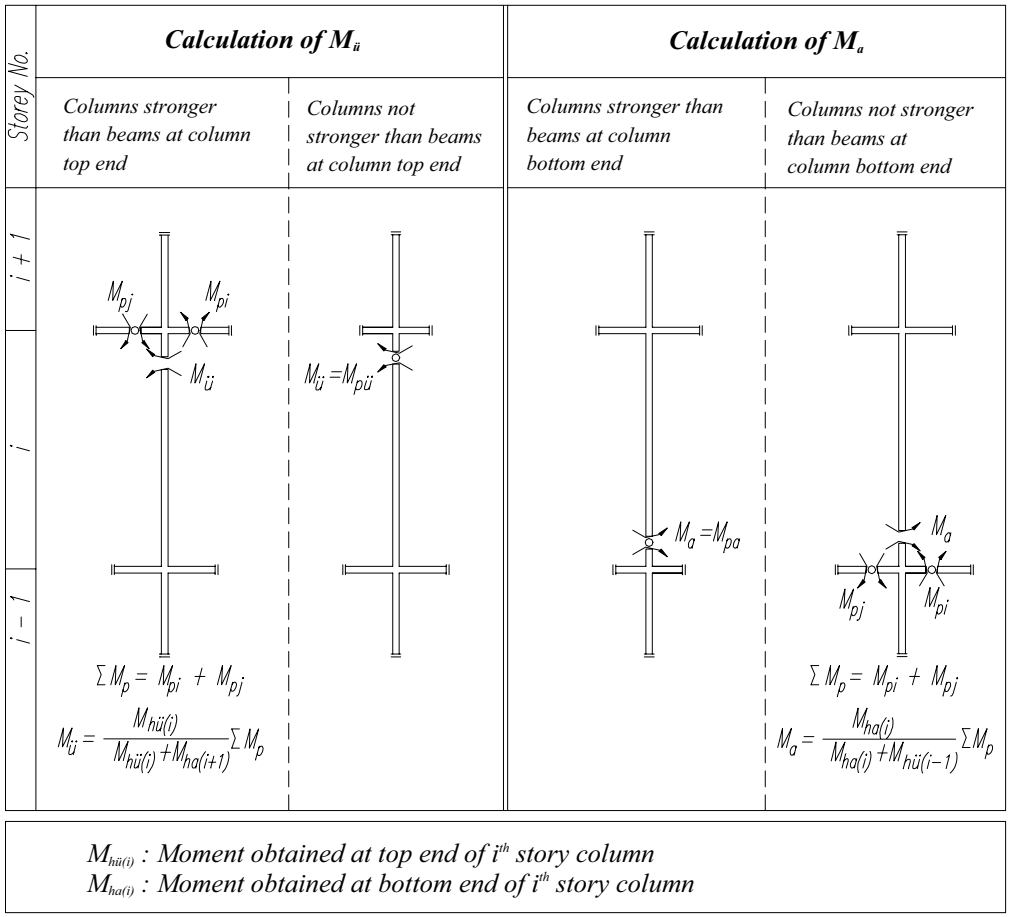

Figure 2.3: M_a and $M_{ü}$ in column shear design

While calculating the shear strength of a reinforced concrete section for a load combination, if this load combination satisfies the following conditions, concrete contribution to the shear strength of the section shall be taken as zero for this load combination:

- The combination shall include earthquake case.
- Shear force due to earthquake case shall be greater than half of the shear force due to whole combination.
- Axial force due to whole combination shall be less than $0.05 A_c f_{ck}$.

2.3.4 Beam-Column Joints

In Turkish Earthquake Code, beam-column joints are divided into two groups as confined and unconfined joints.

- The joints that are surrounded by beams at each side and the width of each beam is

smaller than 75% of the adjoining column width are defined as confined beam-column joints.

- All joints not satisfying the above conditions are defined as unconfined beam-column joints.

The shear force in beam column joints along the earthquake direction considered shall be calculated by the following formula:

$$V_e = 1.25 f_{yk}(A_{s1} + A_{s2}) - V_{col} \tag{2.37}$$

In the case where beams frame into a column from only one side and discontinuous on the other side, A_{s2} shall be taken equal to zero. Moreover, there is a limitation for the maximum shear force at beam-column joints. If the shear force calculated according to Equation 2.37 exceeds the maximum shear force calculated by Equation 2.38 in a beam-column joint, cross-sectional dimensions of columns and/or beams shall be increased and the analysis shall be repeated.

$$\begin{aligned} V_e &\leq 0.60 b_j h f cd \text{ for confined joints} \\ V_e &\leq 0.65 b h f cd \text{ for unconfined joints} \end{aligned} \tag{2.38}$$

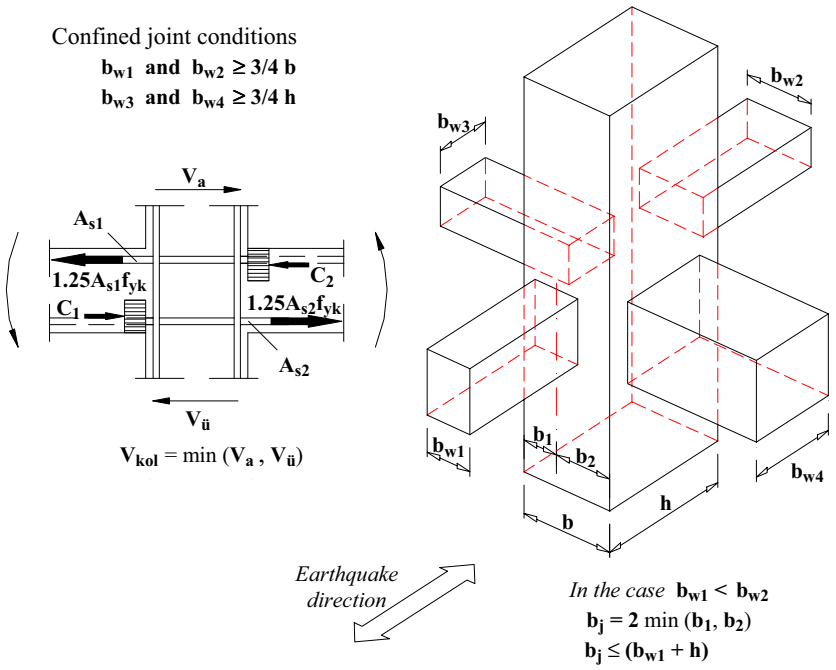

Figure 2.4: Shear force in beam-column joint (TEC-2007)

2.3.4.1 Minimum Transverse Reinforcement in Beam-column Joints

The requirements related to minimum amount, diameter and spacing of transverse reinforcement in beam-column joints are stated as follows:

- Confined beam-column joints
  - The transverse reinforcement shall be at least 40% of the of transverse reinforcement amount computed at the confinement zone of the column below along the height of the joint.
  - The diameter of the transverse reinforcement shall not be less than 8 mm.
  - The spacing of the transverse reinforcement shall not be greater than 150 mm.
- Unconfined beam-column joints
  - The transverse reinforcement shall be at least 60% of the of transverse reinforcement amount computed at the confinement zone of the column below along the height of the joint.
  - The diameter of the transverse reinforcement shall not be less than 8 mm.
  - The spacing of the transverse reinforcement shall not be greater than 100 mm.

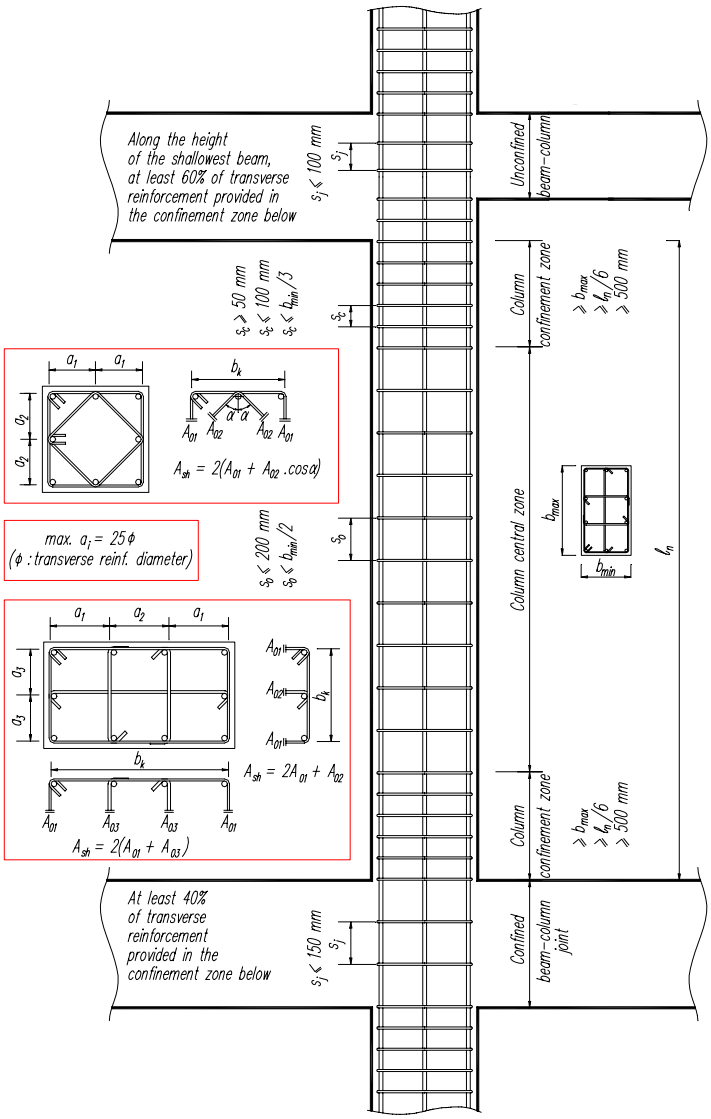

Figure 2.5: Shear Reinforcement Detailing in Columns (TEC-2007)

CHAPTER 3

SECTION ANALYSIS UNDER AXIAL LOAD AND BIAXIAL BENDING MOMENT

3.1 Introduction

In reinforced concrete structural buildings, although most columns are subjected to biaxial bending moment, design is performed according to major bending moment and axial load [Ersoy, 2004]. Especially corner columns, however, are subjected to significant bending moments in two directions and should be designed considering both bending moments.

The fundamental feature of the reinforced concrete section analysis subjected to axial load with biaxial bending moment is that there are two parameters to define the neutral axis, namely depth and inclination of the neural axis. This property makes the analysis more complicated for hand calculations.

In this chapter a procedure developed for the analysis of reinforced concrete sections subjected to axial load and biaxial bending moment is introduced. Having drawn the moment interaction diagram (M_{22} – M_{33}) for specified axial load, it is easy to see whether the design is sufficient or not by locating the design moment in this diagram.

3.2 Sections Under Axial Load and Uniaxial Bending Moment

In Figure 3.1, a reinforced concrete section under axial load and uniaxial bending moment, strain distribution of this section and forces are presented.

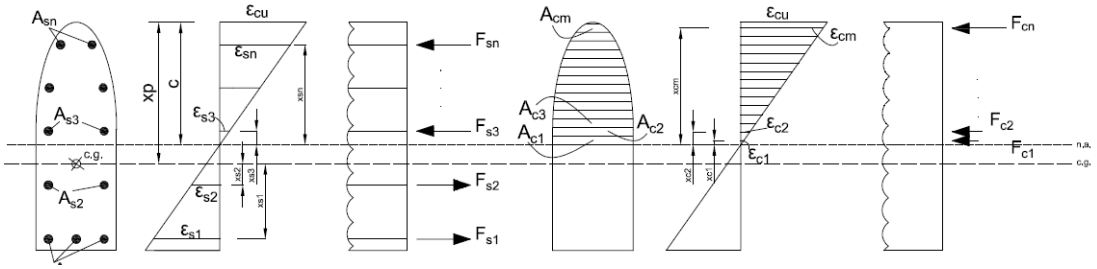

Figure 3.1: A Section Under Axial Load and Uniaxial Bending Moment

In Figure 3.1(a), the steel area is at each level is marked as $A_{s1}, A_{s2}, A_{s3,} \ldots, A_{sn}$. The shaded area is the compression zone of the section. As it can be seen from the Figure 3.1(e), the strain at each level changes linearly from zero to ε_{cu} for compression zone. To calculate the force on compression zone, the compression zone is divided into sections (Figure 3.1(d)). Area of each concrete compression section is marked as $A_{c1}, A_{c2}, A_{c3,} \ldots, A_{cm}$. In Figure 3.1(e), the strain at middle height of each compression slice and distance of each section from the centroid are marked as $\varepsilon_{c1}, \varepsilon_{c2}, \varepsilon_{c3}, \ldots \varepsilon_{cm}$ and $x_{c1}, x_{c2}, x_{c3}, \ldots, x_{cm}$ respectively. The strain in steel in each level is marked as $\varepsilon_{s1}, \varepsilon_{s2}, \varepsilon_{s3}, \ldots \varepsilon_{sn}$. and the distance of each steel layer from the centroid is marked as $x_{s1}, x_{s2}, x_{s3}, \ldots, x_{sn}$ (Figure 3.1(b)). The force at each layer of steel is calculated by multiplying the steel area by the corresponding stress at that level (Figure 3.1(c)). For concrete force, area of each compression section is multiplied by the stress at the mid-height of that section (Figure 3.1(f)). Moreover, "cg" is the center of gravity of the section and "n.a" is the neutral axis.

With above information, the following equilibrium, compatibility, and force deformation equations can be written. Compression and shortening are taken as positive and and the values measured in the direction of eccentricity are also taken as positive.

- Equilibrium :
  - Axial Force Equilibrium :

$$N = \sum(A_{ci}\sigma_{ci}) + \sum(A_{sj}\sigma_{sj}) \tag{3.1}$$

  - Moment About Centroid :

$$M = N * e = \sum(A_{ci}\sigma_{ci}x_{ci}) + \sum(A_{sj}\sigma_{sj}x_{sj}) \tag{3.2}$$

- Compatibility :

From the strain diagram in Figure 3.1(b),

$$\frac{c}{\varepsilon_{cu}} = \frac{x_p - c - x_{sj}}{-\varepsilon_{sj}} \tag{3.3}$$

$$\frac{c}{\varepsilon_{cu}} = \frac{x_p - c - x_{ci}}{-\varepsilon_{ci}} \tag{3.4}$$

- Force Deformation :
  - Bilinear steel model :

$$\sigma_{sj} = \varepsilon_{sj} E_s \leq f_{yd} \tag{3.5}$$

  - Trilinear steel model :

$$\begin{aligned} \sigma_{sj} &= \varepsilon_{sj} E_s && if\ \epsilon_{sj} \leq \varepsilon_{sy} \\ \sigma_{sj} &= \sigma_y && if\ \epsilon_{sy} \leq \varepsilon_{sj} \leq \varepsilon_{sp} \\ \sigma_{sj} &= \sigma_y + \tfrac{\sigma_u - \sigma_y}{\varepsilon_{su} - \varepsilon_{sy}}(\varepsilon - \varepsilon_{sy}) && if\ \epsilon_{sj} \geq \varepsilon_{sp} \end{aligned} \tag{3.6}$$

  - σ_{ci} is determined according to concrete model such as Rectangular Stress Block Model, Rectangular Parabola Model, Hognestad Concrete Model, Kent & Park Unconfined Concrete Model.

There are three equations Equation 3.1, Equation 3.2 and (Equation 3.5 or Equation 3.6) to determine the moment capacity of an arbitrary shaped cross-section. Having known all the geometric and material properties, moment capacity of cross-section is determined for specified axial load, N.

The fundamental feature of the section analysis under axial load and uniaxial bending moment is having the neutral axis parallel to direction of bending. Having known all the geometric properties of the cross-section, arrangement of longitudinal bars, material properties, and design axial load, axial force equilibrium is satisfied by adjusting the depth of neutral axis. Having found the depth of neutral axis, moment capacity is found by just summing moments of all steel and concrete forces about the center of gravity. For different axial load levels, corresponding moment values can be found for a given cross-section. When axial load-moment pairs are plotted, interaction diagram is obtained for that cross-section (Figure 3.2). To check whether the design is sufficient or not, the design axial force-moment pair is plotted on the interaction diagram. If it is in/on the surface, the longitudinal reinforcement is sufficient. Otherwise, the amount of reinforcement should be increased.

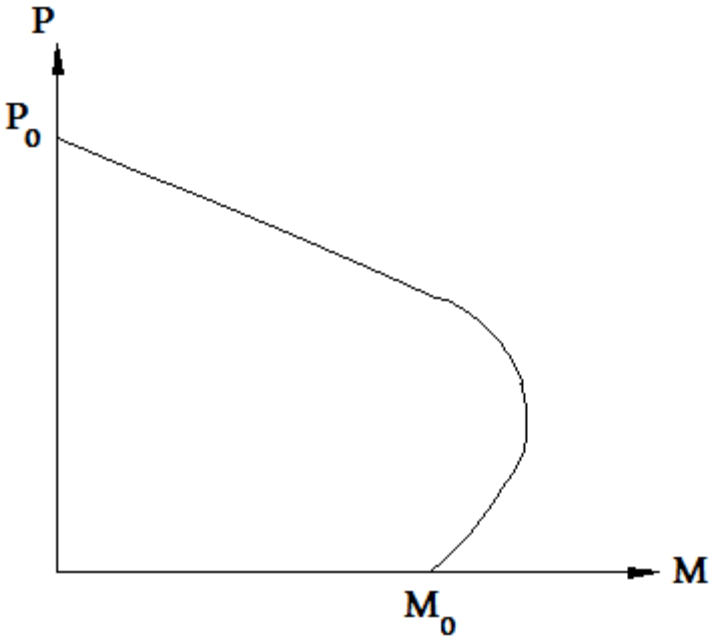

Figure 3.2: Typical Moment Interaction Diagram

3.3 Sections Under Axial Load and Biaxial Bending Moment

In reinforced concrete buildings, most columns are subjected to biaxial bending moment and axial load. In practice, the minor moment in one axis is neglected and columns are designed according to axial load and major uniaxial bending moment. Especially the corner columns, however, might be subjected to axial load and significant bending moments in two principle directions. Neglecting one of these significant bending moments may cause inadequate design. To design such columns, both bending moments in two directions should be taken into consideration.

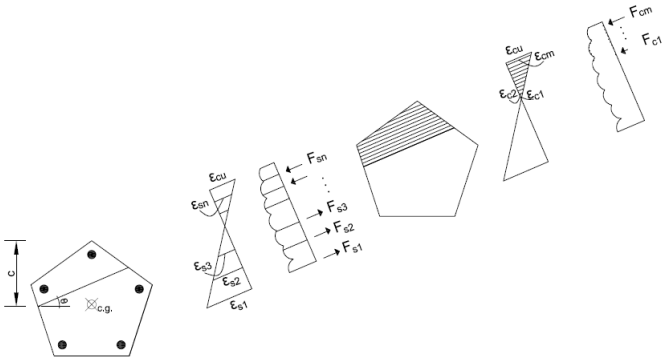

Figure 3.3: A Section Under Axial Load and Biaxial Bending Moment

When the section is under axial load and uniaxial bending moment, the only parameter to define the position of neutral axis is the depth of neutral axis. The depth itself is, however, not enough to determine the position of the neutral axis in case of axial load and biaxial bending. The second parameter to define the position of neutral axis is the inclination of neutral axis (Figure 3.3). These two parameters are changed until the equilibrium is satisfied. However, the solution is difficult and lengthy because of the trial and error approach which includes two variables namely, depth and inclination of neutral axis [Ersoy, 2004]. Moreover, analysis for biaxial bending is significantly more difficult, as moments are not applied in a plane of symmetry [Furlong, 2004].

By varying the depth and inclination of neutral axis, a series of interaction diagrams can be obtained in case of axial load and biaxial bending. When these diagrams are put together, interaction surface (failure surface) is formed (Figure 3.4). For a specified axial load, a horizontal section cut is taken through the interaction surface. This horizontal section cut is a curve which represents the interaction between the two moments, M_{22} and M_{33} (Figure 3.4). For a certain axial load, by checking whether the design moment pair (M_{d22} and M_{d33}) is in/on the taken interaction diagram between M_{22} and M_{33} , the supplied reinforcement can be controlled.

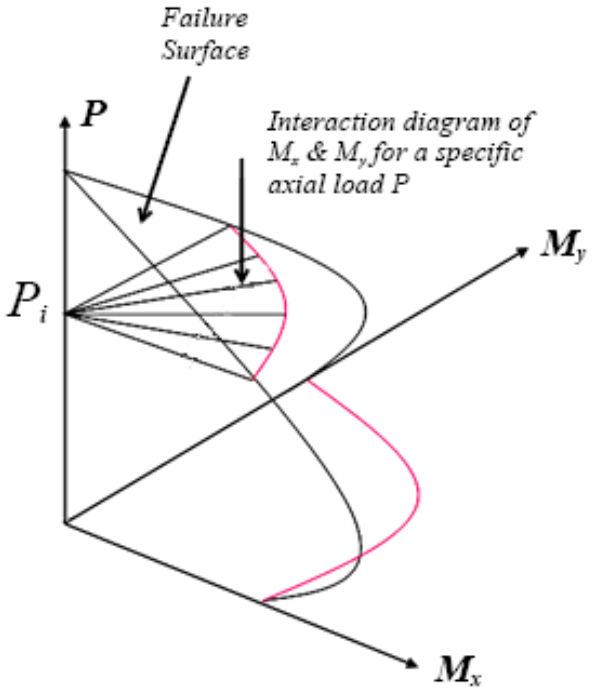

Figure 3.4: Typical Biaxial Interaction Diagram

3.4 Determination of Points on Interaction Surface (Failure Surface)

A procedure to determine points of interaction surface has been developed (Figure 5.8). This procedure is for an arbitrary shaped cross-section and there is no limitation for longitudinal reinforcement mesh in the cross-section. Having defined cross-sectional dimensions, material properties, material models, and reinforcement mesh in the cross-section, the interaction surface of the section can be computed.

The procedure to determine points of interaction surface is outlined below :

- The section is rotated form 0 to $\pi/2$. For each section rotated :
  - Depth of section is calculated.
  - By changing top and bottom strain of returned section, the neutral axis is calculated. For each neutral axis :
    - Steel Forces: Strains at each steel level are calculated. According to calculated strain and preferred steel model, steel force for each longitudinal bar is calculated.
    - Concrete Forces: The compression zone is divided into sections parallel to the neutral axis. The number of division is equal to the depth of neutral axis / strip thickness. Then, the strain at the mid-height of each section is calculated according to similar triangle principle. Having calculated strain at that level, the stress for that level is determined according to concrete model. By multiplying this stress level with strip thickness and strip width, force is obtained for that level. Concrete force is obtained by summing all strip forces in the compression zone.
    - Total Moment: By multiplying all forces (concrete strip forces and steel forces) with their own moment arm according to centroid of the section, moment values are obtained according to rotated section. Moment values about principle directions are obtained by converting moments for rotated section by utilizing the coordinate transformation method described in Appendix B.

Having calculated all points of interaction surface, it is time to determine a horizontal section taken through the interaction surface for design axial load. To do this, all points of interaction

surface are checked and points whose axial component is close enough to design axial force are used to draw interaction diagram between moments.

In determining biaxial interaction diagram, the following assumptions are made :

- Plane sections remain plane after bending.
- Concrete does not take any tension.
- When there is a tension in the cross-section, a constant value of 0.003 is used for ε_{cu}. Under uniaxial compression ε_{cu} is taken as 0.002. ε_{cu} is changed from 0.002 to 0.003 when the whole section is under compression by rotating the strain axis about a fixed point (x_c = 1/3 h) [Ersoy,2004]
- There is a perfect bound between steel and concrete. Therefore, the strain of steel is equal to that of concrete located at the same distance from the neutral axis.

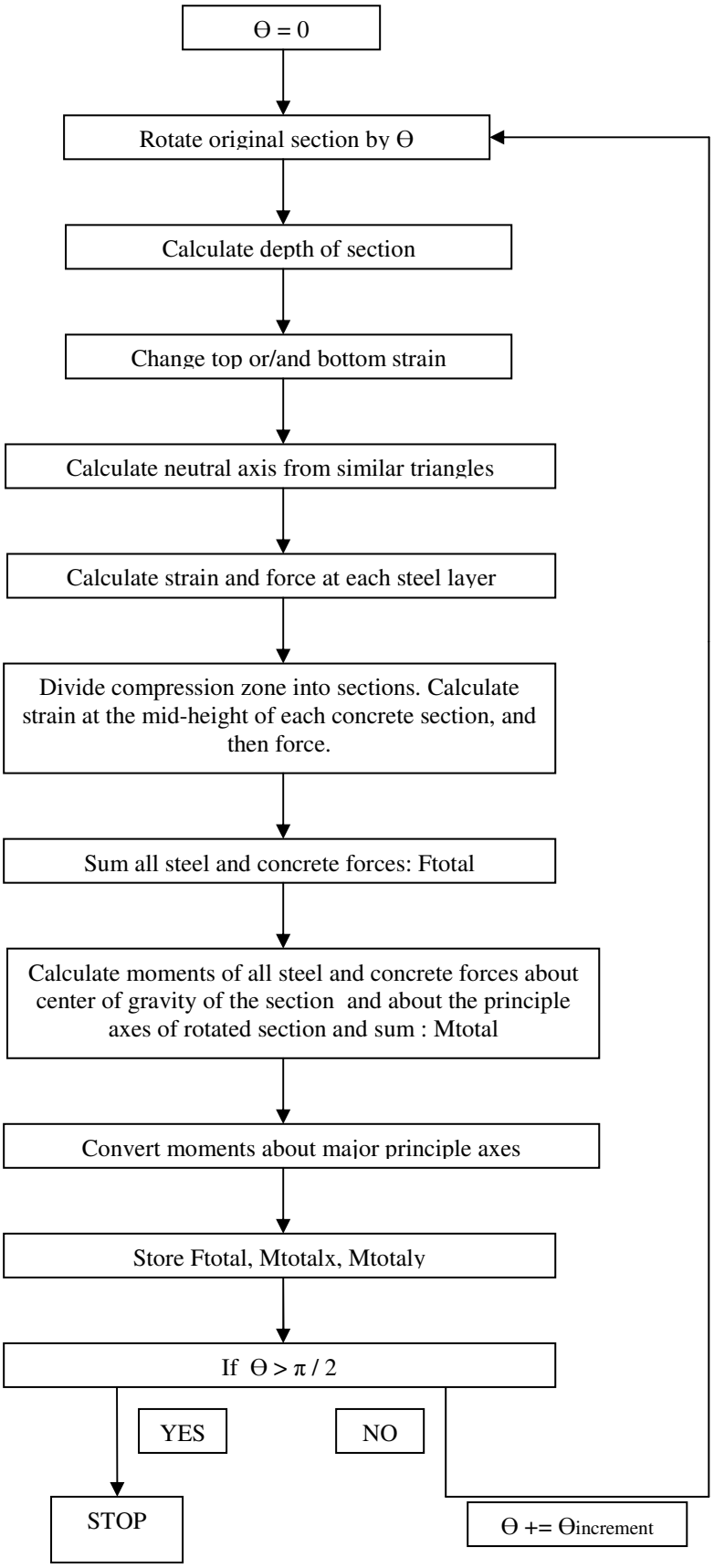

Figure 3.5: Pseudo Code for Calculation of Points of Failure Surface

3.5 An Example Problem

To demonstrate the developed analysis procedure in section 3.4, M_{33} and M_{22} diagram of the cross-section in Figure 3.6 will be drawn for specified design axial load.

Cross-section dimensions, b & h : 300 mm.

Clear cover : 30 mm

Materials : C20 & S420

Longitudinal reinforcement : $4\phi 20$

Design axial load, N_d : 279.5 kN

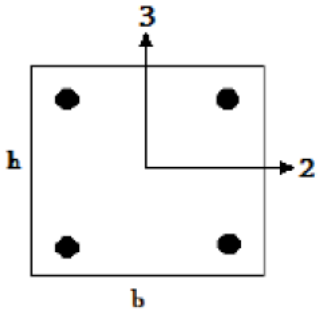

Figure 3.6: An example Problem

As a first step, there is no rotation in cross-section (Figure 3.7). Height of the section is calculated. Then, axial load and moment values are calculated as described in 3.4 by changing top and bottom strain. Since these moments are already about the major axes 2 and 3, there is no need to translate these moments about major axes. The calculated axial load and moment values represent a point on failure surface in biaxial interaction diagram.

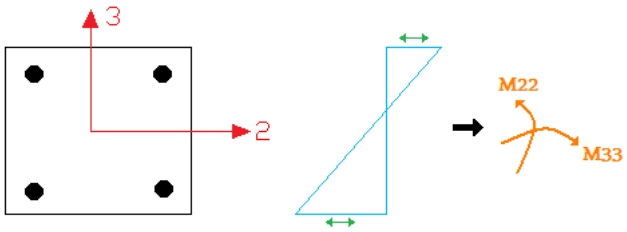

Figure 3.7: Analysis of section rotated by 0^o

Then, the cross-section is rotated by 30^o (Figure 3.8). Having calculated height of the

rotated section, axial load and moment values are calculated according to local axes ($2^{'}$and $3^{'}$) by changing top and bottom strains. Calculated moments ($M_{22}^{'}$ and $M_{33}^{'}$) in local axes ($2^{'}$and $3^{'}$) are converted to moments (M_{22} and M_{33}) in major axes (2 and 3) by utilizing the coordinate transformation method described in Appendix B.

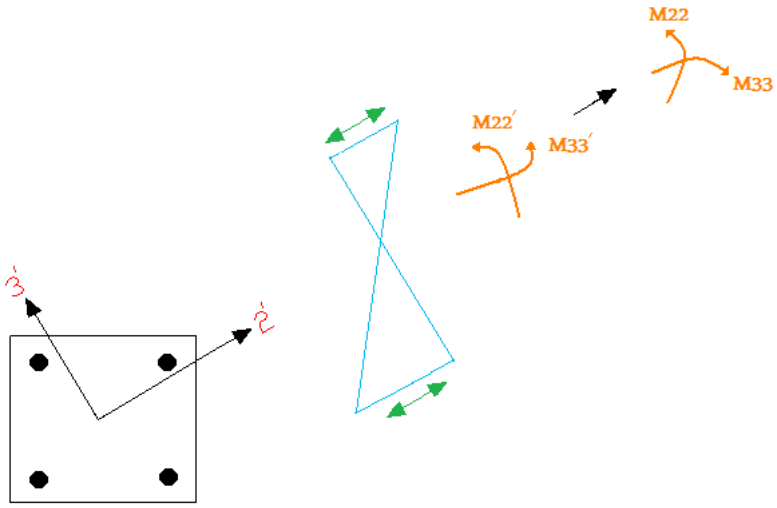

Figure 3.8: Analysis of section rotated by 30^{o}

Next, the cross-section is rotated by 60^{o} and then 90^{o} (Figure 3.9 and Figure 3.10). The same procedure described above for 30^{o} is followed for 60^{o} and 90^{o} and moment values (M_{22} and M_{33}) about major axes are obtained.

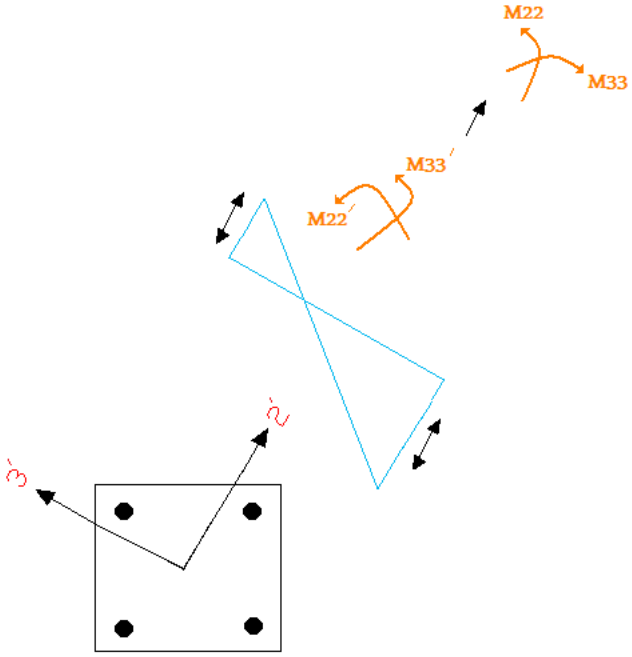

Figure 3.9: Analysis of section rotated by 60^o

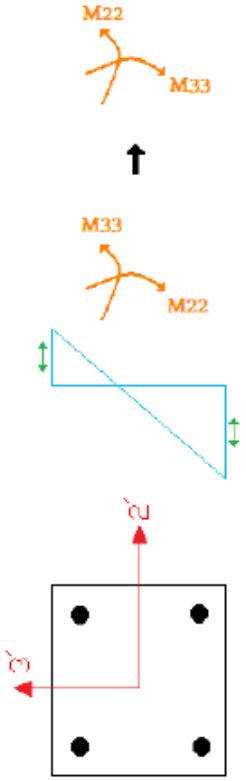

Figure 3.10: Analysis of section rotated by 90^o

When the section rotation is completed, there are lots of points to define the biaxial interaction diagram (failure surface) of the cross-section. Since to use 3D biaxial interaction diagram is difficult to check whether the reinforcement is sufficient or not, a horizontal section cut is taken at the design axial load level. This horizontal section cut is a curve which represents the interaction between the two moments, M_{33} and M_{22}.

Having obtained all biaxial interaction diagram points for the example cross-section, a horizontal section cut is taken through the interaction surface for axial load, 279.5 kN (Figure 3.11).

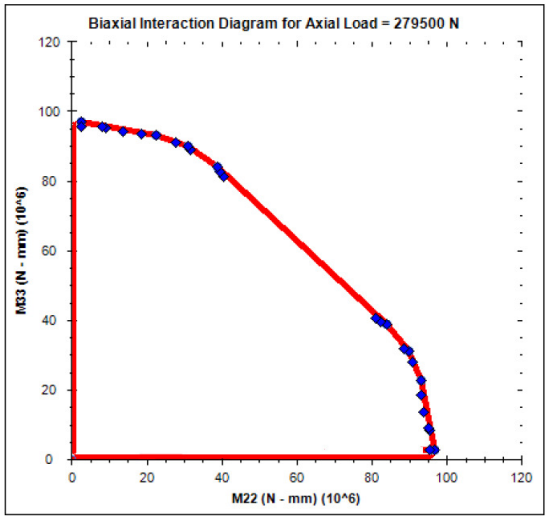

Figure 3.11: M33 - M22 diagram for example section

To obtain a sufficient reinforcement design, the design bending moments, M_{d33} and M_{d22}, should be on/inside the curve in Figure 3.11. For example, if design bending moments M_{d33} and M_{d22} are 80 kN.m and 40 kN.m respectively, the supplied reinforcement is sufficient. If design bending moments M_{d33} and M_{d22}, however, are 80 kN.m and 70 kN.m respectively, the reinforcement amount is not enough to resist against design moments.

CHAPTER 4

DATA STRUCTURE

4.1 Introduction

In a reinforced concrete building, there are various subparts that constitute a structure, such as stories, columns, beams, column lines, beam strips and reinforcements. All these parts have common and special characteristics. To reflect reinforced concrete building in a data model, all properties of these subparts should be transferred in the data model in detail.

In this chapter, an object oriented data model which is developed for characterizing a reinforced concrete building and facilitating code based design and detailing tasks in a multi-user environment is presented.

4.2 General Structure

The design process of a reinforced concrete building can be briefly summarized in four steps, i.e, preliminary design, structural modeling and analysis, design and detailing of reinforcement. These steps are usually followed in an ordered fashion at the initial stages but as the design progress, this process becomes an iterative one with continuous repetitions of analysis and design steps. Moreover, each step pictures the same building in different point of views, where a structural model is composed of frame elements but the design model utilizes structural components, such as columns and beams.

Thus, in order to support the iterative process of design and coordinate the different information requirements of each step, information related to design process is considered into two layers : Design Layer and Analysis Layer. Figure 4.1 presents the general architecture of the data model that is composed of two layers with the developed four main packages : Analysis Package, Design Entities Package, Reinforcement Package and Design Code Pack-

age. Analysis Package was introduced for defining the geometry and section properties of members, and for storing the analysis results of the structure obtained from the structural analysis step. Design Entities Package contains the information about the structural entities of a structure such as beam, column, joint, and story definitions. Detailed geometry, properties and relations of reinforcing bars in reinforced concrete frames were defined in the Reinforcement Package. Design Code Package was developed to facilitate calculations and checks according to various design codes.

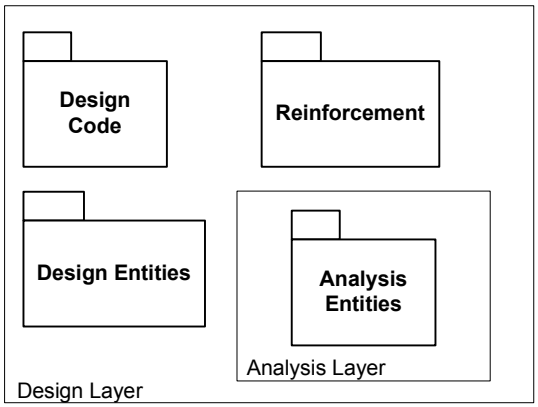

Figure 4.1: General architecture of the object library

4.2.1 Analysis Package

In case of linear structural analysis, there are no distinction between beams and columns but there are one dimensional frame elements and nodal points for modeling frame structures. Therefore, in order to represent these frame elements and their end nodes, *FrameElement Class* and *Node Class* were introduced. The cross-sectional properties of each FrameElement object were stored at *FrameCrossSection Class*. The analysis results and displacement values of various locations of the frame elements were stored at certain points called Station. Relationships of classes introduced in the Analysis Package are presented in Figure 4.2.

4.2.1.1 FrameElement Class

While modeling a structure in an analysis program, frame elements are utilized for defining columns or beams between nodes (Figure 4.3). These frame elements are the basic elements of the structure. In this study, *FrameElement Class* is introduced to represent these basic parts of the structure. The major properties of a frame element are the section, start and end nodes, type of the frame element (whether it is a part of a beam or a column) and

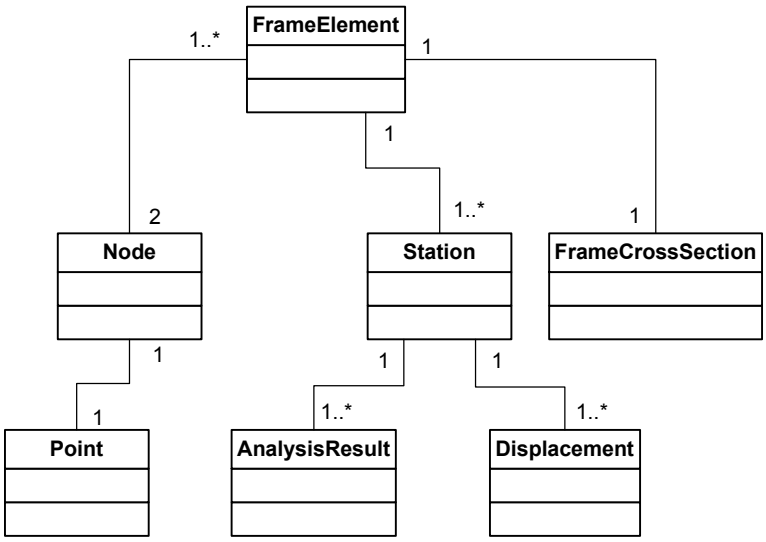

Figure 4.2: Analysis Package

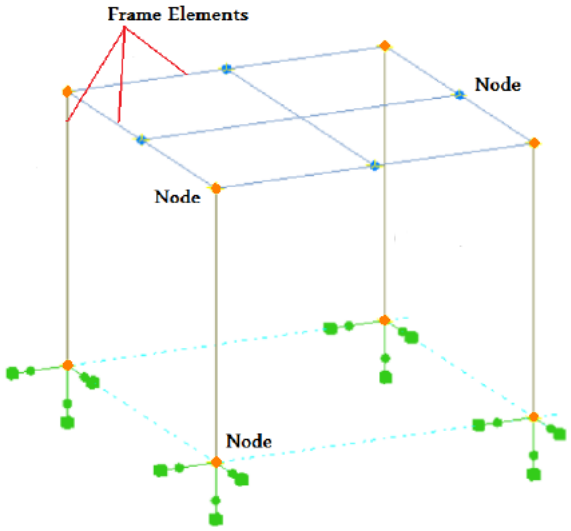

Figure 4.3: Major Elements In a Structure

stations on the frame element (Figure 4.2 & Figure 4.4). To reflect all these properties on *FrameElement Class*, the following additional classes are introduced : *FrameCrossSection*, *Node* and *Station*.

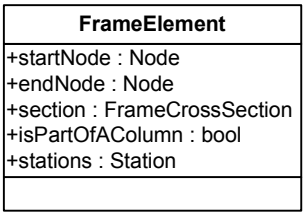

Figure 4.4: Attributes in FrameElement Class

4.2.1.2 FrameCrossSection Class

To design and detail a column, cross-sectional properties of the column should be known. These properties are kept in *FrameCrossSection Class*. The required parameters related to cross-section of a column are width, depth, area, moment of inertia about the major axis (I_{33}) and minor axis (I_{22}), section modulus about the major axis (S_{33}) and minor axis (S_{22}), radius of gyration about the major axis (i_{33}) and minor axis (i_{22}) (Figure 4.5). These cross-sectional properties of a column are utilized to design the cross-section under axial load and biaxial bending moment, to design the column under shear force or to determine the arrangement of longitudinal reinforcements at beam-column joints.

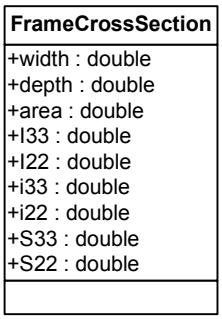

Figure 4.5: Attributes in FrameCrossSection Class

4.2.1.3 Node Class

Frame elements are defined between two nodes. Therefore, a frame element is connected to two nodes, namely start and end node. On the other hand, more than one frame element may connect to a single node. To illustrate, in Figure 4.6 although all frame elements have two nodes at ends, there are 6 frame element connecting to a common node. It means that the "node A" has 6 frame element, 4 of which are horizontal and others are vertical.

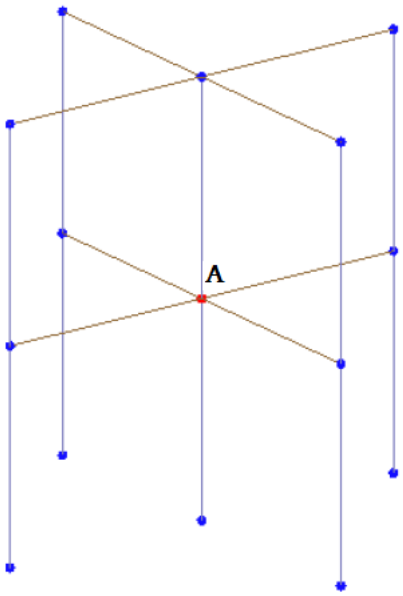

Figure 4.6: Frame elements connecting to a single node

In a *Node Class*, coordinates of the node, frames connecting to this node and nodal displacements are the major parameters (Figure 4.7). Therefore, *Point Class* (Figure 4.8) and *Displacement Class* (Figure 4.9) are introduced to represent the coordinates of a node and the displacement at a node respectively.

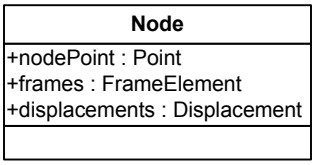

Figure 4.7: Attributes in Node Class

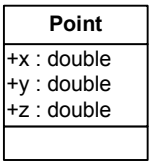

Figure 4.8: Attributes in Point Class

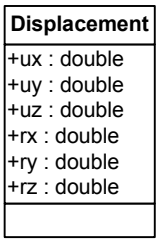

Figure 4.9: Attributes in Displacement Class

4.2.1.4 Station Class

In order to store the analysis results and displacements for each load case at various locations on a frame element, *Station Class* (Figure 4.10) was introduced. The number and location of station points are determined during the modeling of the structure and the internal forces and displacements at these points are computed by the analysis program. The distance between a station point and start node of the frame element is stored as the location of the station point.

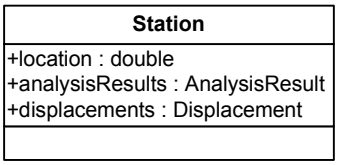

Figure 4.10: Attributes in Station Class

4.2.1.5 AnalysisResult Class

Having modeled the structure in a structure analysis program and introduced different kinds of load cases on the structure, analysis is performed and internal forces of each frame element at station points are obtained. To store these internal forces for each load case at each station point, *AnalysisResult Class* (Figure 4.11) is introduced.

In *AnalysisResult Class,* axial force, shear force in local x and local y direction, torque, and bending moments about local x and local y directions are stored.

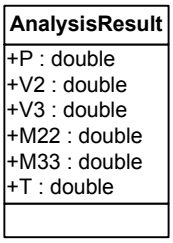

Figure 4.11: Attributes in AnalysisResult Class

4.2.1.6 LoadCase Class

Having modeled a reinforced concrete structure in an analysis program, different types of loads are applied to the structure. In order to define these loads, *LoadCase Class* is introduced. Each load case is defined with two strings, one of them is the name of load case and the other is the type of the load case such as "DEAD" or "LIVE" (Figure 4.12).

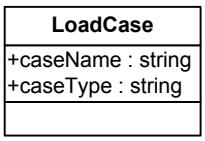

Figure 4.12: Attributes in LoadCase Class

4.2.1.7 LoadCombination Class

The design codes require that the design forces to be used for the calculation of reinforcement in reinforced concrete structures are determined with the superposition of different load cases. Thus, in order to define a loading case formed with the superposition of different load cases, *LoadCombination Class* is introduced (Figure 4.13 and Figure 4.14). Since the multipliers of each load case in a load combination can be different form each other, an extra class is required to define a load case with its multiplier in a load combination. Because of this reason, *CombinationParams Class* is introduced (Figure 4.15).

LoadCombination Classes are created by the Code Class and stored in Building Class.

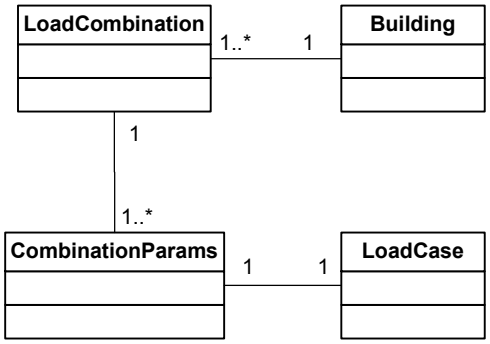

Figure 4.13: Class Relationship of Building Class

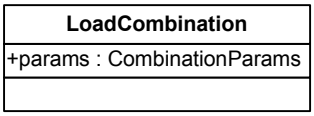

Figure 4.14: Attributes in LoadCombination Class

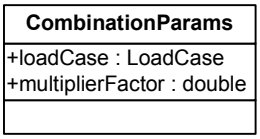

Figure 4.15: Attributes in CombinationParams Class

4.2.2 Design Entities Package

The geometrical definitions of the classes belonging to the Analysis Package are not adequate for designing a reinforced concrete structure. Realization of an actual structure's geometry during the design calculations is different than that of the analysis step. At the analysis step, the structure is modeled with one-dimensional frame members connected to two different nodes. These nodes may not always coincide with the actual beam or column connections, as in the cases where frame members are fine meshed to provide continuity with the meshing of the slabs. In such cases, beams of a frame can be modeled with several frame elements. On the other hand, due to the monolithic nature of cast-in-place reinforced concrete members, their design is performed system based. In other words, continuous beams are often designed as a whole. Similarly, columns on the same grid points are designed and detailed continuously from the foundation level to the top of the building. Moreover, the design of beam-column joints belonging to special moment frames have special seismic de-

sign requirements, hence these connections must be specially attributed. In order to provide such improved definitions to the structural engineer, specific classes were defined. Design Entities Package contains higher level geometrical objects, which groups, connects, or utilizes the objects in the Analysis Package. Relations between these classes are presented in Figure 4.16.

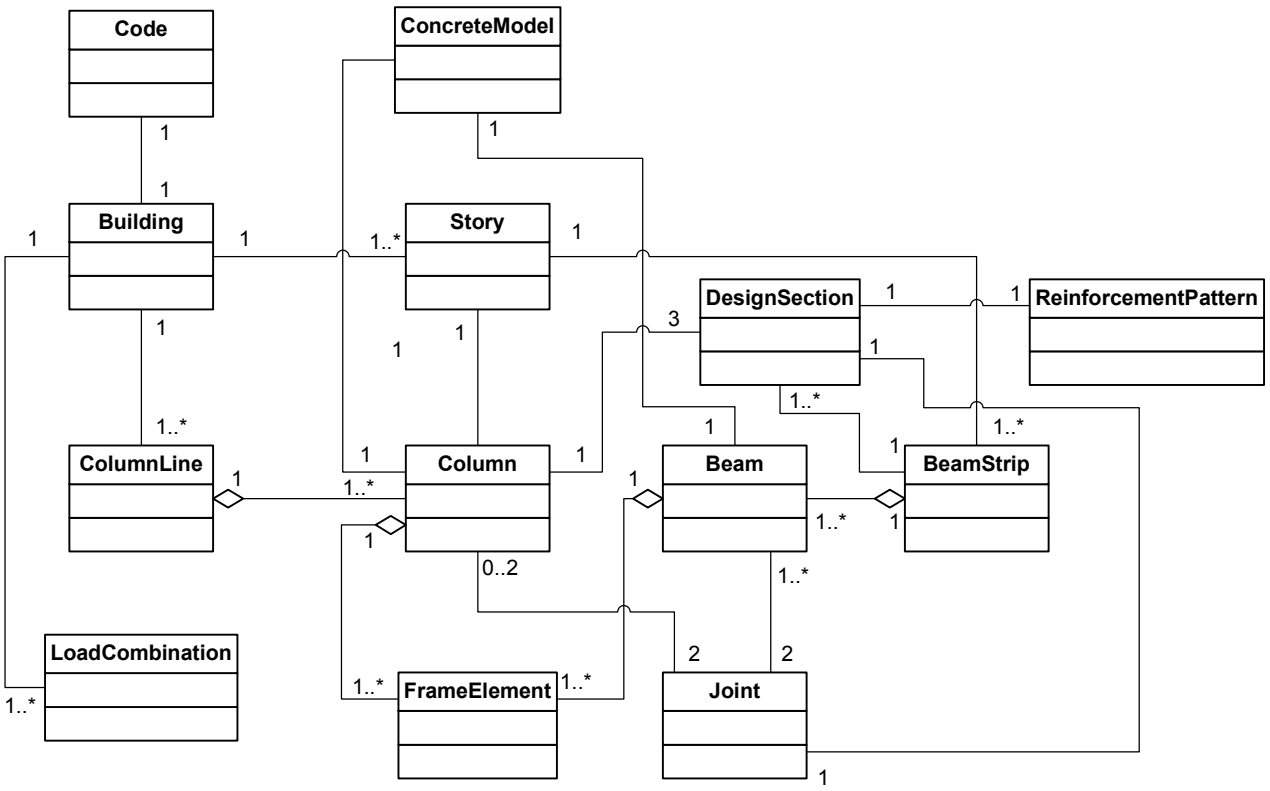

Figure 4.16: Design Entities Package

4.2.2.1 Column Class

Columns are defined as vertical structural components between two stories. While modeling a reinforced concrete building, a column can be defined with more than one frame elements (Figure 4.17). In such cases, there are more than one frame element for a column in the list "frames" defined in section 4.2.2.5 and that's why frame elements that constitute the same column should be detected for each story.

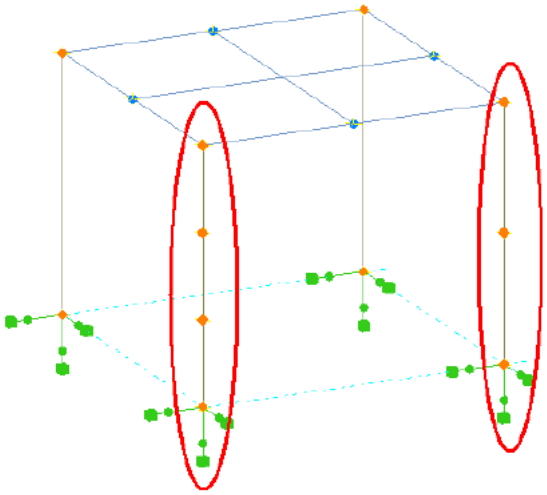

Figure 4.17: Columns may be entered as more than one piece

All detected frame elements construct a new column defined with *Column Class* (Figure 4.18 and Figure 4.19). These frame elements are stored in a list called frame_list in such a way that the frame element at the bottom of the column is the first element, the one at the top is the last element of the list.

The defined column must reflect all properties of its frame elements. In other words, cross-section of column should be the same as its frame elements. Therefore, *Column Class* includes *a FrameCrossSection Class* for storing cross-sectional information. Moreover, in order to define material properties of the column, the *Column Class* stores a *ConcreteModel Class*.

In a reinforced concrete column, there are usually two types of reinforcement, namely longitudinal reinforcement and transverse reinforcement. To design reinforcement and to store information related to reinforcement, three *DesignSection Classes* are defined per *Column Class* for top, middle and bottom confinement zones. In addition, start and end joint are the other parameters in *Column Class*.

Column
+frame_List : FrameElement +startJoint : Joint +endJoint : Joint +section : FrameCrossSection +designSections : DesignSection +concModel : ConcreteModel

Figure 4.18: Attributes in Column Class

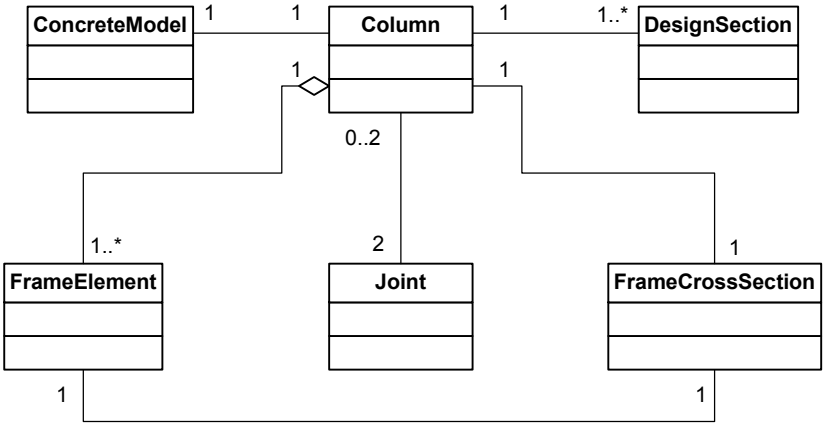

Figure 4.19: Class Relationship of Column Class

4.2.2.2 Beam Class and BeamStrip Class

Beam Class is defined as a design member that connects to two *Joint* objects and composed of a single or series of horizontally positioned *FrameElement* objects. *BeamStrip Class* is created to enable design of continuous beams. *BeamStrip Class* stores an ordered array of *Joint* and *Beam Classes* on itself. This way, the design algorithms can move between Beam and Joint objects to perform calculations such as calculating the design moments and the amount and the total length of the longitudinal reinforcement at the beam-column joints [Anil, 2009].

4.2.2.3 Joint Class

Node Classes where more than one *FrameElements* from different planes are connected to are defined as *Joint Classes*. Therefore, beams, which are composed of horizontally positioned *FrameElements*, and columns, which are composed of vertically positioned *FrameElements*, connecting to a common node are stored in a *Joint Class*. Moreover, in order to check the shear strength of beam-column joint and store possible transverse reinforcements at beam-column joint, a single DesignSection Class was defined as a part of the Joint Class. Figure 4.20 presents the relationship of Joint Class with other classes.

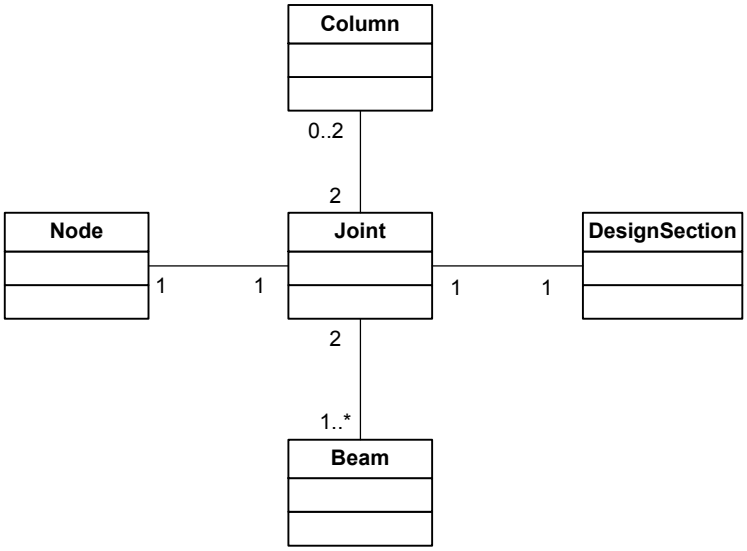

Figure 4.20: Class Relationship of Joint Class

4.2.2.4 ColumnLine Class

In practice columns are not designed for a single story, but for the whole height of the building. In other words, columns at the same grid points are designed and detailed as a whole. Thus, in order to store information about columns through the height of the building, *ColumnLine Class* (Figure 4.22 and Figure 4.23) was introduced. Two *ColumnLine* objects of an example building are presented in Figure 4.21 with dashed lines.

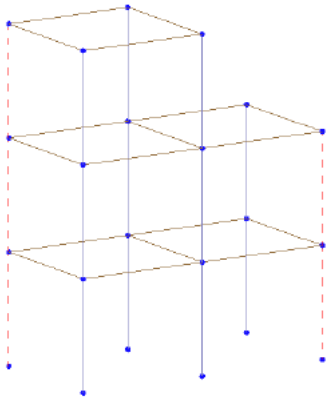

Figure 4.21: Examples of Column Line

Having detected all columns in a story, columns at the same grid point at each story are detected with end node connectivity information and added to the column_list of *Column-Line Class*. In *a ColumnLine Class*, the beginning and end joints of the column line are also

stored for facilitating the preparation of column reinforcement detail drawings.

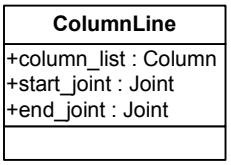

Figure 4.22: Attributes in ColumnLine Class

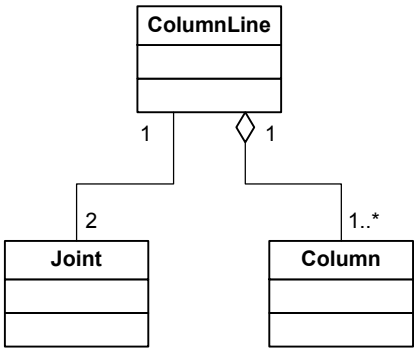

Figure 4.23: Class Relationship of ColumnLine Class

4.2.2.5 Story Class

Node, FrameElement, Beam, BeamStrip and Column objects that are at the same story level are grouped into a *Story Class*. Node and FrameElement objects are stored in the *Story Class* during the initialization of the project and then Beams, Columns and BeamStrips are created by the algorithms of the *Story Class*. While investigating the slenderness effects in columns and calculating the moment magnification factors, the story drifts and story heights must be known in addition to the total vertical load and horizontal story shear (TS500_2000) in order to determine whether frames are sway or non-sway.

In *Story* Class, there are two boolean parameters to reflect whether the story is sway or non-sway in x and y direction : isSwayX and isSwayY. Height and Elevation are other required parameters to classify the story as sway or non-sway and to check the lateral deflection of the story in x and y direction. A *Story Class* also keeps information related with nodes, frames, beams, beam strips and columns located at the story (Figure 4.24 & Figure 4.25).

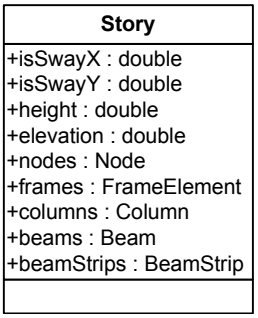

Figure 4.24: Attributes in Story Class

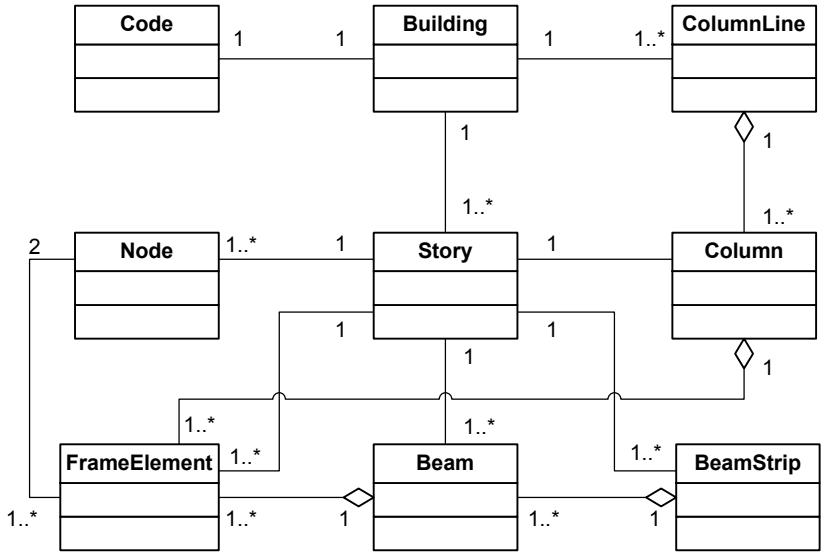

Figure 4.25: Class Relationship of Story Class

4.2.2.6 Building Class

While designing and detailing of reinforced concrete structures, some general parameter related to the structure such as code parameters, loading parameters, story information and columnLines are usually required. Therefore, to store these general information related to the reinforced concrete structure to be designed, an outer class, namely *Building Class,* was introduced (Figure 4.26).

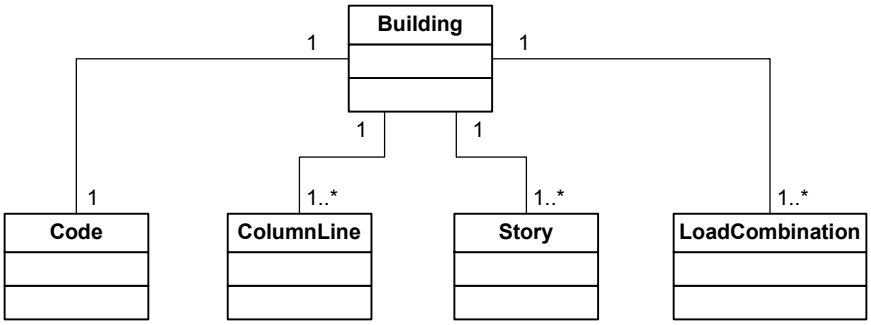

Figure 4.26: Class Relationship of Building Class

4.2.2.7 DesignSection Class

Longitudinal reinforcement in a reinforced concrete column exists for the whole height of the column and the amount of longitudinal reinforcement is determined by designing the top and bottom face of the column. Moreover, there are closely spaced transverse reinforcement at column ends for seismic design of columns which requires shear design at more than one section. Therefore, in order to perform design calculations and to store reinforcement at a particular section, *DesignSection Class* was introduced (Figure 4.27).

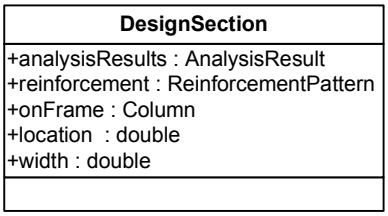

Figure 4.27: Attributes in DesignSection Class

In order to design a *DesignSection*, analysis results at this section are required. That's why a *DesignSection* stores a hashtable of *AnalysisResult Classes.* Having designed the section, details of longitudinal and transverse reinforcement at this section are kept in *Reinforcement-Pattern Class* stored in *DesignSection Class.* In order to define the region of a *DesignSection* , two parameters, namely location and width are stored in the *DesignSection Class.* Moreover, a *DesignSection Class* keeps the owner column or beam with the parameter called "OnFrame".

DesignSection Classes are utilized either for flexural or shear design. For flexural design, they represent a section. For shear design, however, they represent a region defined by the width and location of the region. The same transverse reinforcement spacing and pattern are utilized at a single DesignSection. In case of column design, three *DesignSection Classes* are defined per column for top, middle and bottom confinement zones. This way, different reinforcement information for each region can easily be stored and reached.

4.2.2.8 MaterialModel Class

Material properties of concrete and steel such as modulus of elasticity of concrete, compressive strength of concrete, modulus of elasticity of steel and yield strength of steel are used while designing and detailing of columns and beam-column joints. In order to define material properties of concrete and steel, *Concrete Model Class* and *SteelModel Class* derived from *MaterialModel Class* were introduced, respectively. Moreover, *Concrete Model Class* stores an enumerator for the concrete model that defines the strain-stress relationship of concrete. According to the value of this enumerator, Rectangular Stress Block, Hognestad or Kent & Park models were utilized while computing the interaction diagram for columns. Likewise, *SteelModel Class* stores an enumerator for the steel model that defines the strain-stress relationship of steel.

In order to perform design and detailing of the members modeled with different concrete strengths, a relation between *ConcreteModel Class* and *Beam* and *Column Classes* was created. Moreover, a relation between *SteelModel Class* and *Rebar Class* which is the abstract class of LongitudinalBar and TranverseBar Classes is created to allow having reinforcements with different steel strengths (Figure 4.28).

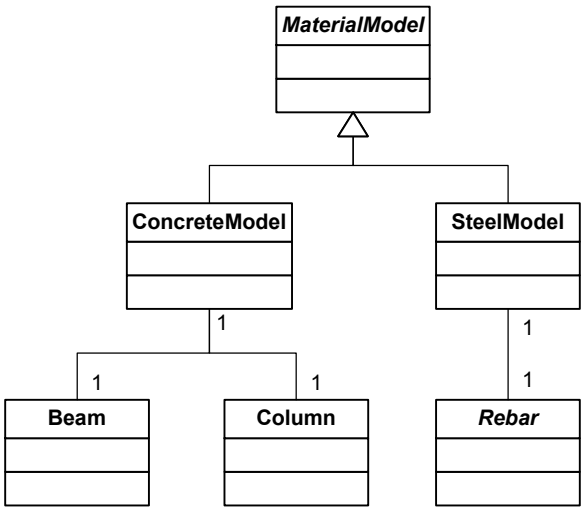

Figure 4.28: Class Relationship of MaterialModel Class

4.2.3 Reinforcement Package

In order to define the detailed geometry and design specific information of reinforcing bars in reinforced concrete frames, several classes were developed in the Reinforcement Package. First, the *Rebar Class* was introduced to define a base class for reinforcing bars. Object relations in the Reinforcement Package are presented in Figure 4.29. Two child classes were introduced to define more specialized properties for the two kinds of reinforcement found in reinforced concrete members : LongitudinalBar and TransverseBar. *LongitudinalBar Class* represents the reinforcement that is placed parallel to the member axis, whereas TransverseBar objects are placed perpendicular to the member axis.

The longitudinal bars can be placed in a section in many. Thus, in order to keep all the LongitudinalBar objects in a section and facilitate the design, *LongitudinalPattern Class* was defined. In a similar manner, *TransversePattern Class* stores the TransverseBar objects in each section at a region. *EndStyle Class* was introduced for defining the geometry of the longitudinal bar ends. Each LongitudinalBar object has two EndStyle objects one for beginning, one for the end of the bar.

According to the proposed object model, each DesignSection object has a single ReinforcementPattern object. The *ReinforcementPattern Class* not only stores one LongitudinalPattern and one TransversePattern object belonging to its DesignSection but also has methods for modifying the placement of longitudinal and transverse reinforcement in relation with each other.

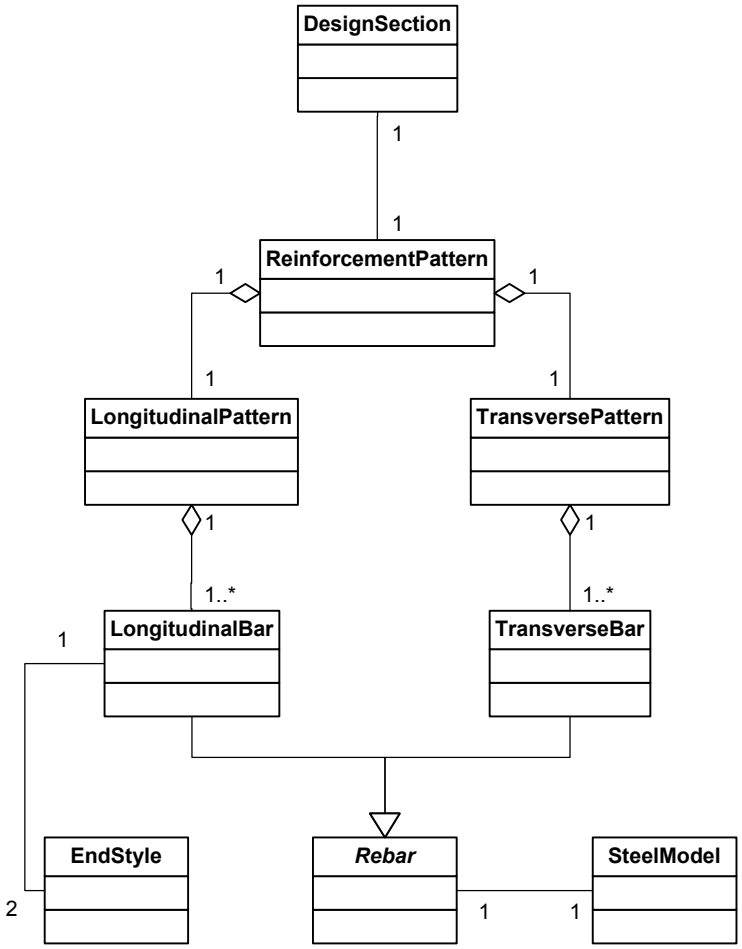

Figure 4.29: Reinforcement Package

4.2.3.1 ReinforcementPattern Class

In reinforced concrete columns there are two types of reinforcement: longitudinal and transverse reinforcement. To keep all information related to reinforcement in a cross-section, *Reinforcement Pattern Class* (Figure 4.30) was introduced. In *Reinforcement Pattern Class* there are two main classes which are *Longitudinal Pattern Class* for storing the location and sizes of longitudinal reinforcement and *Transverse Pattern Class* for storing the size, orientation, and spacing of transverse reinforcement.

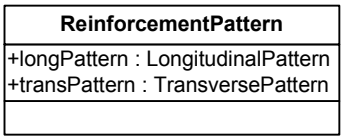

Figure 4.30: Attributes in ReinforcementPattern Class

To keep diameter and surface texture of each reinforcement, whether longitudinal, transverse or miscellaneous, *Rebar Class* (Figure 4.31) was introduced. There are two types of surface texture of reinforcement, namely Plain and Deformed. Surface texture is kept in *Rebar Class* as enumerator (Table 4.1) and is utilized during the calculation of development length of the longitudinal reinforcement. Moreover, *Rebar Class* includes a method named as *Area()* that returns the area of a single reinforcement.

Table 4.1: Surface texture enumerator

enumSurfaceTexture
plain
deformed

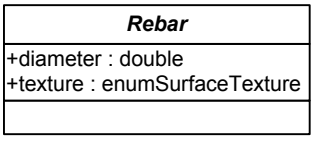

Figure 4.31: Attributes in Rebar Class

4.2.3.2 LongitudinalBar Class

LongitudinalBar is the data which is used to keep information related to longitudinal bars in a reinforced concrete column and is inherited from *Rebar Class* (Figure 4.31). Basically, a *LongitudinalBar* was defined from the middle height of the column to the lower face of the joining beam (Figure 4.32). This basic part of a *LongitudinalBar* is represented with two points stored in a list in *LongitudinalBar* (Figure 4.33). In addition to this main part which is always straight, the geometry of beam-column joints, development length and splice location determine the bottom and top portion of the longitudinal reinforcement. To illustrate, there are two alternatives for bottom portion of the longitudinal reinforcement presented in Figure 4.32. The first alternative is that bottom portion extends to the floor level if the splice is done at the floor level. The second alternative for the bottom portion of

the longitudinal reinforcement is that bottom portion extends with an amount of half of the development length if the splice is done at mid-height of the column. Moreover, there are four alternatives for the top portion of selected longitudinal reinforcement in Figure 4.32. The first two alternatives are performed by extending longitudinal reinforcement into the upper column. In remaining alternatives, however, longitudinal reinforcement does not extend into the upper column but extends to the existing beam or turns into itself.

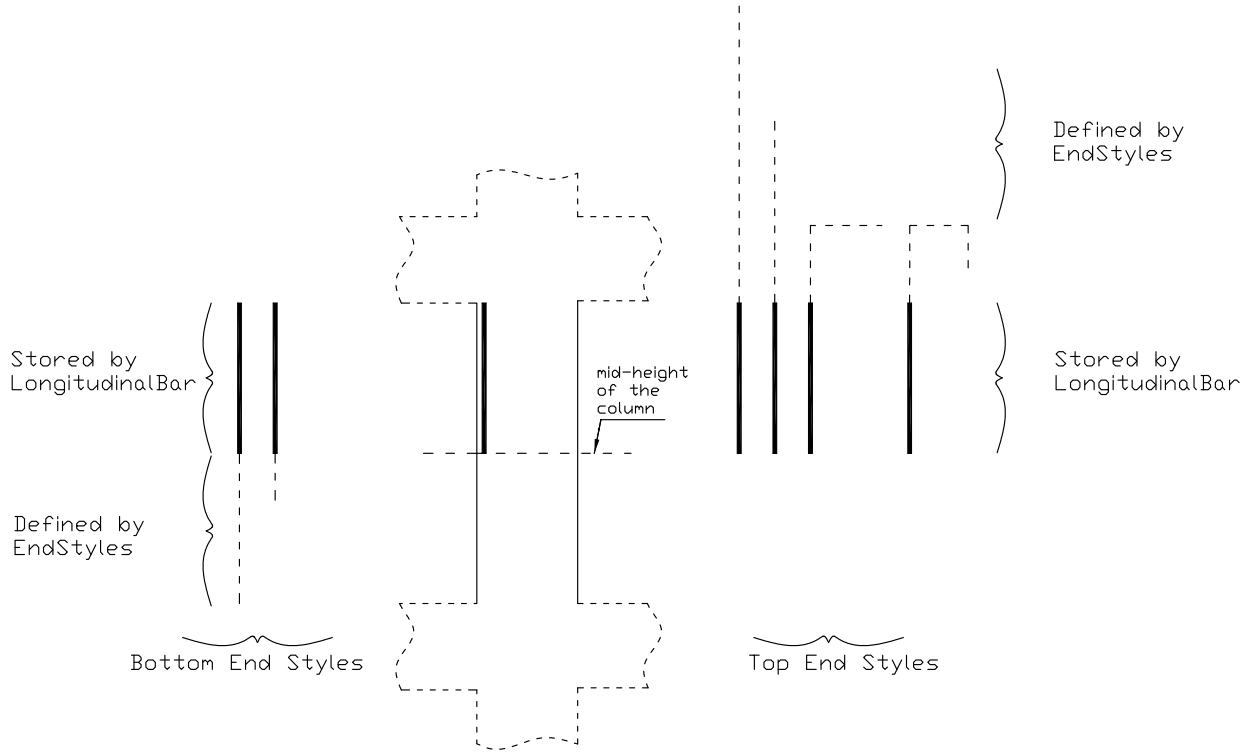

Figure 4.32: Basic Straight Part of a LongitudinalBar and Possible Alternatives

If the dimensions of bottom and top column at the beam-column joints are different from each other or there is no column above the top joint, the longitudinal reinforcements coming from bottom column do not go straight up, but they are directed to beam if exists or turn into itself. In order to describe the shape of the longitudinal reinforcements at such locations, *EndStyle Class* (4.2.3.3) was introduced. Since bottom and top beam-column joints may differ in geometry for a column, two *EndStyle Classes* are defined for a single longitudinal reinforcement to represent its shape at these zones.

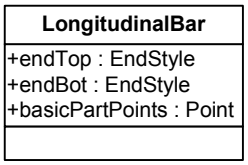

Figure 4.33: Attributes in LongitudinalBar Class

4.2.3.3 EndStyle Class

There are three types of lap-splice shapes at beam-column joints at Turkish Earthquake Code. If both top and bottom column exist at a joint and dimensions of top and bottom columns are the same, the longitudinal bars coming from the bottom column can go straight into the upper column with an additional length determined according to the development length and splice region specifications of design codes. If the reinforcement is spliced at the floor level, this additional length is equal to beam depth plus the development length of the reinforcement calculated according to specified design code (Figure 4.34).If it is spliced at mid-height of the column, however, the additional distance is equal to beam depth plus the half of the clear length of the column plus half of the development length (Figure 4.35). To represent this vertical length in *EndStyle* Class, an entity whose name is *ver_Main* is introduced.

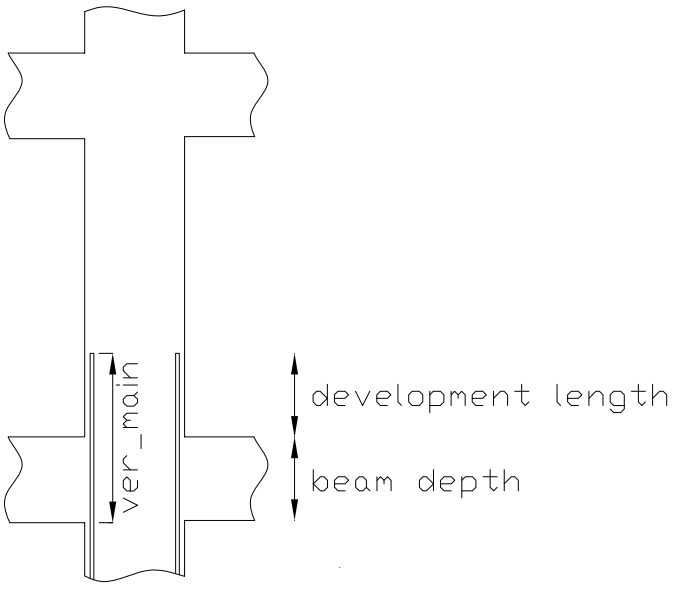

Figure 4.34: Splice is done at the floor level

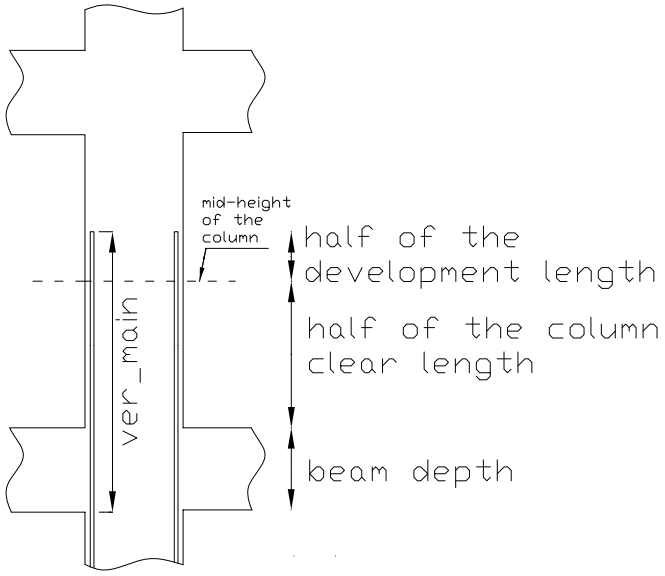

Figure 4.35: Splice is done at the mid-height of the column

The second type of the longitudinal reinforcement shape at the beam-column joints is defined when the dimensions of bottom and top column are different or top column does not exist (Figure 4.36). In this case, the longitudinal reinforcement coming from the bottom column does not go straight into the upper column but it is directed to the other side slab or beam whose width is greater than the dimension of column in that direction. In this case, *ver_Main* by itself is not sufficient to describe the shape of the longitudinal reinforcement. Therefore, an extra parameter is required to describe this horizontal part of the longitudinal reinforcement at beam-column joints. This connection parameter was defined as *hor_Main* and describes the horizontal length of the bent portion (Figure 4.36).

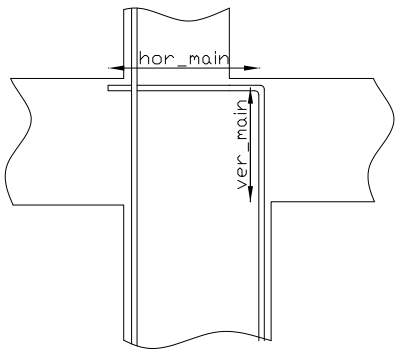

Figure 4.36: Second type of the end-style shape

When the dimensions of the top and bottom columns are not the same or top column does not exist and there is no slab or beam whose width is greater than the dimension of bottom column on one of the sides of the column and if the sum of the lengths of *ver_Main* and *hor_Main* is smaller than the development length calculated according to design code, the longitudinal reinforcement should be bent down into the bottom column (Figure 4.37). To represent this part, *ver_Sec* parameter is introduced in *EndStyle* Class. This parameter is determined according to the design code and development length.

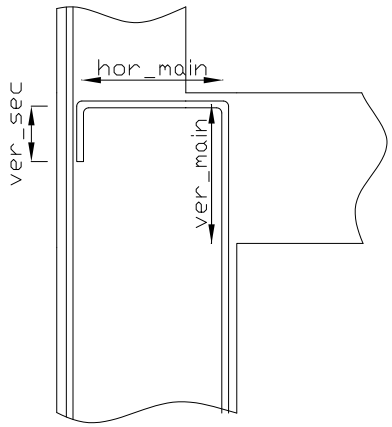

Figure 4.37: Third type of the end-style shape

In all of the cases described above, bottom portion of the longitudinal reinforcement coming from top column to the bottom column are always straight and its length is determined according to the splice region and development length, too. In other words, only "ver_main" parameter of *EndStyle* is calculated for bottom portion of the longitudinal reinforcement. If the splice is done at the floor level, this distance is equal to the half of the clear length of the column (Figure 4.38). If the splice is done at the mid-height of the column, however, this length is equal to the half of the development length (Figure 4.39).

Another parameter in *EndStyle Class* is the direction of the bent reinforcement, since longitudinal reinforcement at different edges of a column are bent at different directions (Figure 4.40). Thus, in order to store the bent direction of the longitudinal reinforcements, an integer representing positive x direction, negative x direction, positive y direction and negative y direction is used.

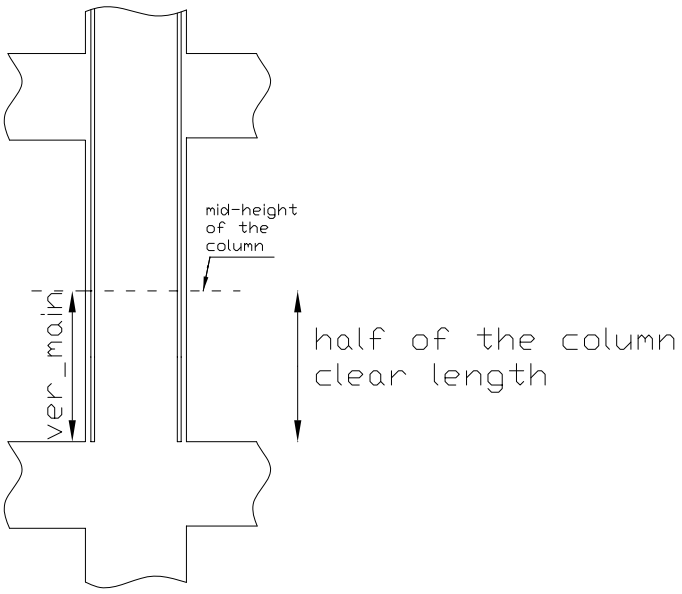

Figure 4.38: Splice is done at the floor level

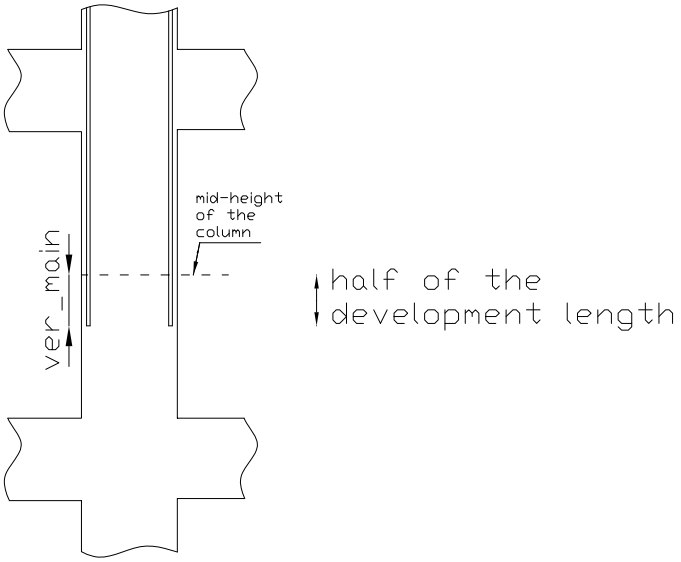

Figure 4.39: Splice is done at mid-height of the column

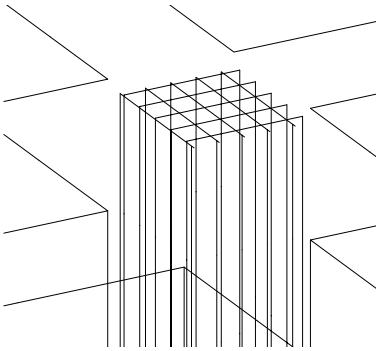

Figure 4.40: Different Directions of Longitudinal Reinforcement at Beam-Column Joints

A single longitudinal reinforcement has two end styles : top and bottom. In order to distinguish these two *End Style Classes*, a boolean *isTop* is stored in *EndStyle Class* (Figure 4.41).

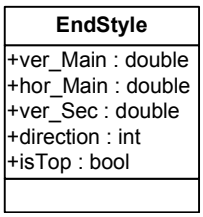

Figure 4.41: Attributes in EndStyle Class

4.2.3.4 LongitudinalPattern Class

Longitudinal Pattern Class (Figure 4.42) is the main class that stores all information related to longitudinal reinforcement in a concrete column cross-section. First of all, the arrangement of longitudinal bars in the cross-section should be known. In other words the exact location of each longitudinal bars should be defined to perform strain, stress and force calculations. Therefore, an enumerator defined in section 4.2.3.5 related to longitudinal reinforcement pattern is the first parameter *Longitudinal Pattern Data* stores. In addition to the arrangement of longitudinal bars in a cross section, the pyhsical properties of each longitudinal bars should be known. These properties are the diameter, surface texture, points of path of a bar and top and bottom end style of a bar. All these properties are stored in *LongitudinalBar Class*. Thus, a list of *LongitudinalBar Class* is the second parameter in *Longitudinal Pattern Class* (Figure 4.29).

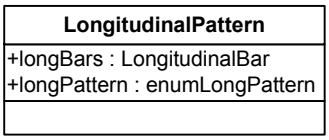

Figure 4.42: Attributes in LongitudinalPattern Class

4.2.3.5 Pre-defined Longitudinal Reinforcement Patterns

While designing column longitudinal reinforcement, location of each longitudinal bar is used to calculate strain, stress and force on longitudinal bars. Common longitudinal patterns are defined in order to determine location of longitudinal bars in a column cross-section (Figure 4.43). In these pre-defined longitudinal patterns, there are certain number of longitudinal bars and the location of each longitudinal bar is determined according to width and height

of the column-cross section and clear cover. Each pre-defined longitudinal reinforcement pattern is called with an enumerator (enumLongPattern).

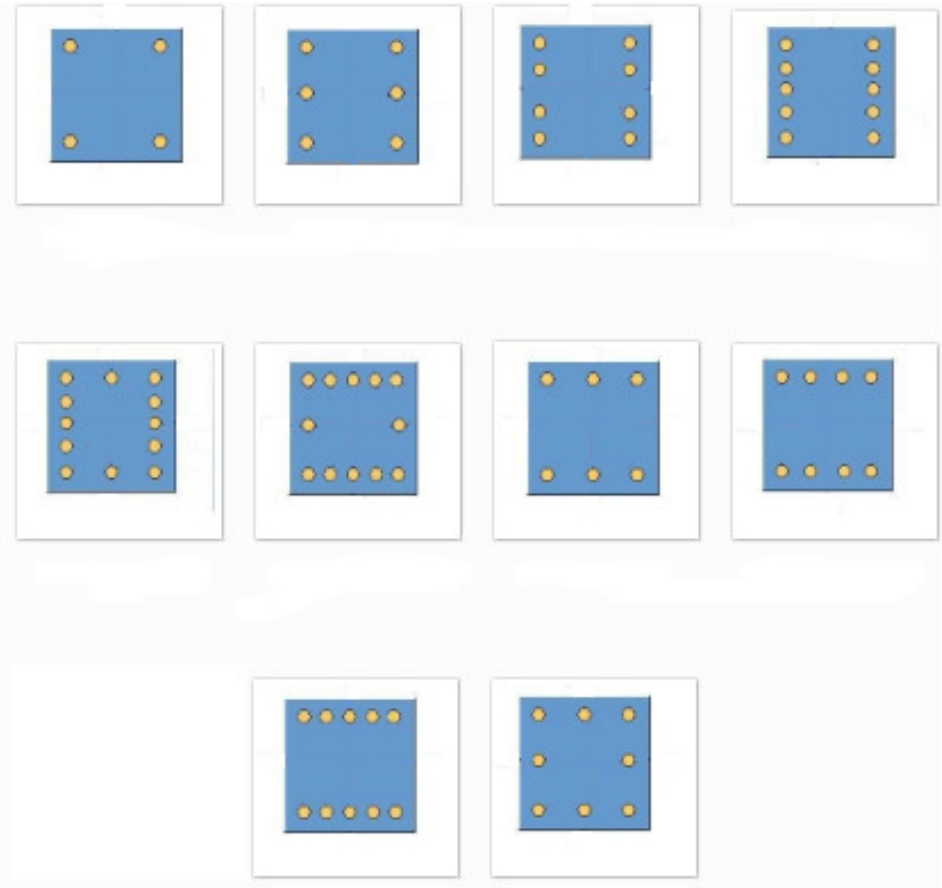

Figure 4.43: Pre-defined Longitudinal Reinforcement Patterns

According to the selected pre-defined longitudinal pattern, number of longitudinal bars and location of each longitudinal bar in the cross section are determined with the parameters of height and width of the cross-section and clear cover.

4.2.3.6 TransverseBar Class

TransverseBar Class (Figure 4.44) is defined in order to store information related to transverse bars in a column and is derived from *Rebar Class*. In addition to the properties defined in the *Rebar Class*, a *TransverseBar Class* keeps a list of points that defines the shape of the transverse reinforcement.

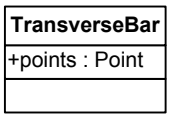

Figure 4.44: Attributes in TransverseBar Class

4.2.3.7 TransversePattern Class

In a reinforced concrete column, there are three zones which are top confinement, bottom confinement and mid zone. Especially in seismic regions, different amount of transverse reinforcement are placed in these zones. Thus, in order to keep number, distance and diameter of Transverse Bars in these zones, *Transverse Pattern Class* (Figure 4.45) is introduced.

To define the location of each transverse bar in a concrete cross-section and calculate shear reinforcement area, an enumerator defined in section 4.2.3.8 is stored as the first parameter in *Transverse Pattern* Class. A list keeping *TransverseBar* in the concrete cross-section is the second parameter in *Transverse Pattern* Class. For the whole height of the column, spacing, number of transverse reinforcements and distance of first transverse reinforcement to the start joint are other parameters related to transverse reinforcement in a reinforced concrete column.

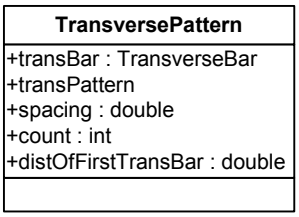

Figure 4.45: Attributes in TransversePattern Class

4.2.3.8 Pre-defined Transverse Reinforcement Patterns

In order to determine the location of transverse reinforcement bars in a concrete cross-section and to determine the number of shear reinforcement legs in two major directions and therefore area of the shear reinforcement, pre-defined transverse reinforcement patterns are utilized in shear design of reinforced concrete columns. (Figure 4.46). Each pre-defined shear reinforcement pattern is called with an enumerator (enumTransPattern).

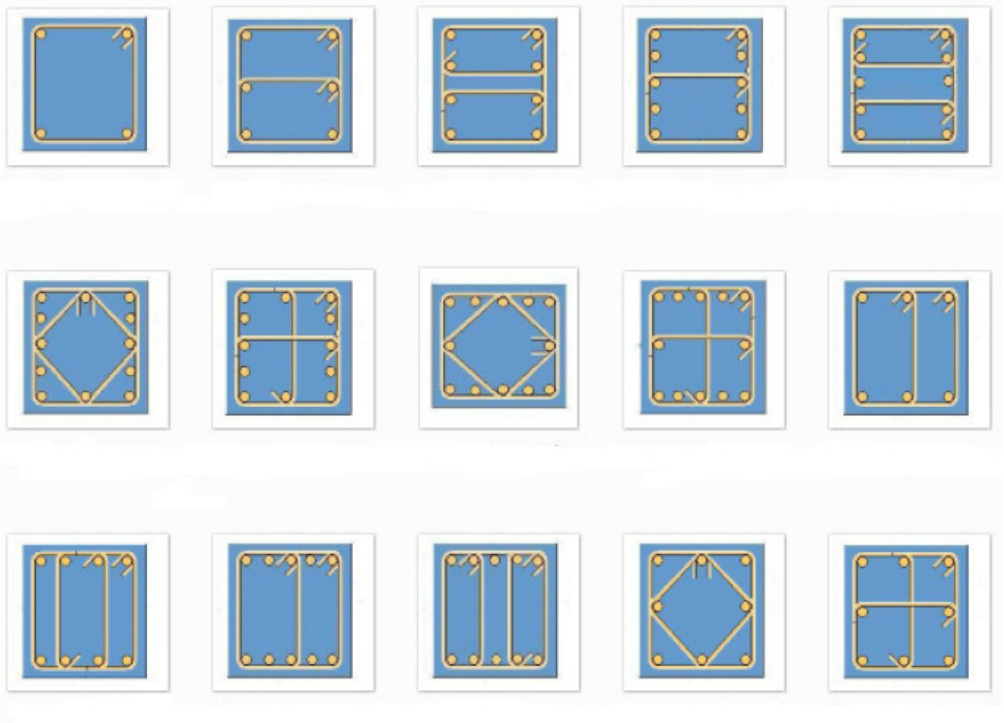

Figure 4.46: Pre-defined Transverse Reinforcement Patterns

4.2.4 Design Code Class

Any structural design must follow specific rules defined by the design codes. Design codes contain minimum member size and reinforcement ratio requirements, strength limitations, material properties, behavior definitions, load combinations, design methods, strength equations, serviceability requirements and detailing rules. Thus, in order to perform design and detailing of reinforced concrete columns and beam-column joints according to a design code, *Code Class* is introduced.

In the *Code Class,* various overwritten methods that describe the common features of the design codes were defined. These overwritten methods utilized during the design and detailing of reinforced concrete columns and beam-column joints are presented in Figure 4.47.

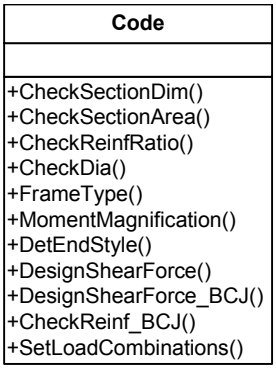

Figure 4.47: Overwritten Methods in Code Class

- *CheckSectionDim()* : Compares cross-section dimensions of a column with the minimum cross-section dimensions specified by design codes.
- *CheckSectionArea() :* Compares cross-section area of a column with the minimum cross-section area specified by design codes.
- *CheckReinfRatio() :* Compares longitudinal reinforcement ratio in a column cross-section with maximum and minimum longitudinal reinforcement ratios specified by design codes.
- *CheckDia() :* Compares longitudinal and transverse reinforcement diameter in a column cross-section with maximum and minimum diameters specified by design codes.
- *FrameType() :* Checks whether frames in a story are sway or non-sway.
- *MomentMagnification() :* Calculates moment magnification factor for a column.
- *DetEndStyle() :* Determines detailing shape of a longitudinal bar at top and bottom joints of the column.
- *DesignShearForce()* : Calculates the design shear force for a reinforced concrete column specified by design codes.
- *DesignShearForce_ BCJ()* : Calculates the design shear force for a beam-column joint specified by design codes.
- *CheckReinf_ BCJ()* : Compares amount, spacing and diameter of reinforcement at a beam-column joint with the requirements specified by design codes.
- *SetLoadCombinations()* : Combines load cases to obtain most unfavorable loading conditions according to design codes.

The *Code Class* is defined as the main class that provides methods common to all design codes, but the design equations and rules of different design codes are usually not the same even if they follow the similar design approach. Because of this reason, the classes representing the country codes are inherited from *Code Class* and the methods are overriden according to predefined functionality of each method presented in Figure 4.47.

In Turkish reinforced concrete construction, two different codes are utilized : TS500_2000 and TEC (Turkish Earthquake Code). TS500_2000 defines the general rules and specifications for reinforced concrete structures whereas TEC focuses on the earthquake resistant design. In other words, the design against all loadings other than earthquake loads must be performed according to TS500_2000 and the design for earthquake loads should follow TEC. Thus, in order to handle dual codes during the design, two child classes, one for TS500_2000 and one for TEC, were introduced. All the design equations and specifications for both codes were implemented within these two classes and managed by the higher level *TurkishCodes Class* (Figure 4.48). The methods specific to both TS500_2000 and TEC are presented in Figure 4.49.

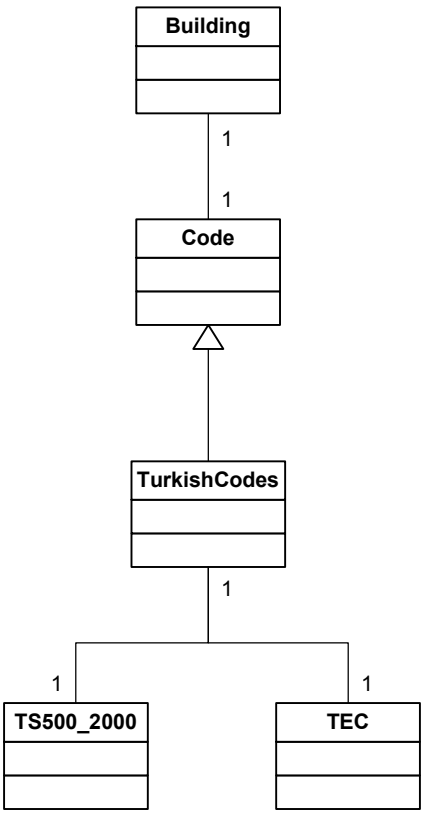

Figure 4.48: Class Relationship of Code Class

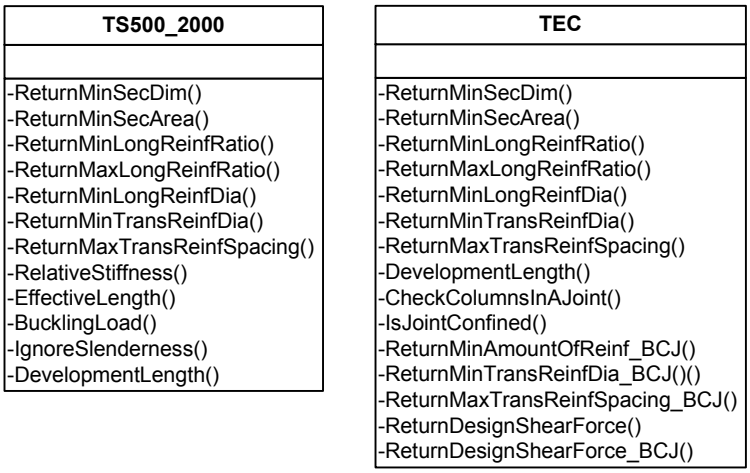

Figure 4.49: Methods in TS500_2000 and TEC

The detailed descriptions of each method in *TS500_2000 Class* are as follows :

- *ReturnMinSecDim()* : Returns the minimum column cross-section dimensions defined in TS500_2000 (Section 2.2.1).
- *ReturnMinSecArea()* : Returns the minimum column cross-section area defined in TS500_2000 (Section 2.2.1).
- *ReturnMinLongReinfRatio()* : Returns the minimum longitudinal reinforcement ratio in a column cross-section defined in TS500_2000 (Section 2.2.1).
- *ReturnMaxLongReinfRatio()* : Returns the maximum longitudinal reinforcement ratio in a column cross-section defined in TS500_2000 (Section 2.2.1).
- *ReturnMinLongReinfDia()* : Returns the minimum longitudinal reinforcement diameter in a column cross-section defined in TS500_2000 (Section 2.2.1).
- *ReturnMinTransReinfDia()* : Returns the minimum transverse reinforcement diameter in a column cross-section defined in TS500_2000 (Section 2.2.1).
- *ReturnMaxTransReinSpacing()* : Returns the maximum transverse reinforcement spacing in a RC column defined in TS500_2000 (Section 2.2.1).
- *RelativeStiffness()* : Calculates the relative stiffness value of a joint (Section 2.2.2.1). "I_{22}" or "I_{33}" parameters of *Frame Cross Section* that *Column* and *Beam* include and length of *Column* and *Beam* are used to calculate relative stiffness value of a *Joint*.

- *EffectiveLength()* : Calculates the effective length of a column (Section 2.2.2.1). To decide whether a *Column* is sway or not, "isSwayX" and "isSwayY" parameters of a *Story* that the *Column* belongs to are used. Next, α values (relative stiffness values) for top and bottom joints of the *Column* are calculated with the *RelativeStiffness()* method of *TS500_2000 Class*. Then, effective length multiplier, k, is calculated depending on sway properties of the *Column*.

- *Buckling Load()* : Calculates the buckling load of a RC column (Section 2.2.2.2). Effective length of a Column (l_k) is calculated with *EffectiveLength()* method of *TS500_2000 Class*. To decide whether a *Column* is sway or not, "isSwayX" and "isSwayY" parameters of a *Story* that the *Column* belongs to are used. If the *Column* belongs to a non-sway frame, design sustained axial load (N_{gd}) and total design axial load (N_d) are computed from the element forces of the *Column* for each combination in the *Building Class*. If the *Column* belongs to a sway frame, however, the sum of the design shear forces caused by the sustained load (V_{gd}) and sum of the design shear forces (V_d) are computed from the element forces of all *Columns* in the *Story* that the *Column* belongs to for each combination in the *Building Class*. Then, the resultant buckling load (N_k) is the minimum one among the loads calculated for each combination.

- *IgnoreSlenderness()* : Checks whether slenderness is neglected for a RC column or not (Section 2.2.2.4). To decide whether a *Column* is sway or not, "isSwayX" and "isSwayY" parameters of a *Story* that the *Column* belongs to are used. The result of *EffectiveLength()* method of *TS500_2000 Class* is used as effective length of a *Column*, l_k. Moreover, "i_{22}" and "i_{33}" parameters of *Frame Cross Section* (Figure 4.5) that *Column* includes are used as the radius of gyration, i.

- *DevelopmentLength()* : Calculates development length of longitudinal reinforcement described in specified design code (Section 2.2.3). To calculate development length, "diameter" parameter of *Longitudinal Bar* (Figure 4.31) and "fyd" parameter of *Steel Model* stored in *Longitudinal Bar* (Figure 4.29) are used.

The detailed descriptions of each method in *TEC Class* are as follows :

- *ReturnMinSecDim()* : Returns the minimum column cross-section dimensions defined in TEC (Section 2.3.1).

- *ReturnMinSecArea()* : Returns the minimum column cross-section area defined in TEC (Section 2.3.1).

- *ReturnMinLongReinfRatio()* : Returns the minimum longitudinal reinforcement ratio in a column cross-section defined in TEC (Section 2.3.1).

- *ReturnMaxLongReinfRatio()* : Returns the maximum longitudinal reinforcement ratio in a column cross-section defined in TEC (Section 2.3.1).

- *ReturnMinLongReinfDia()* : Returns the minimum longitudinal reinforcement diameter in a column cross-section defined in TEC (Section 2.3.1).

- *ReturnMinTransReinfDia()* : Returns the minimum transverse reinforcement diameter in a column cross-section defined in TEC (Section 2.3.1).

- *ReturnMaxTransReinSpacing()* : Returns the maximum transverse reinforcement spacing in a RC column defined in TEC (Section 2.3.1).

- *DevelopmentLength()* : Calculates the development length of the longitudinal reinforcement described in specified design code (Section 2.3.1). To calculate development length, "diameter" parameter of *Longitudinal Bar* (Figure 4.31) and "fyd" parameter of *Steel Model* stored in *Longitudinal Bar* (Figure 4.29) are used.

- *ChecksColumnsInAJoint()* : Checks whether *Columns* in a *Joint* are stronger than *Beams* in that *Joint* or not (Section 2.3.2). While calculating moment resistance of a *Column* or a *Beam*, *Longitudinal Reinforcement Pattern* in *Reinforcement Pattern* (Figure 4.30) that a *Column* or a *Beam* includes is used to obtain longitudinal reinforcement mesh in cross-section, *Frame Cross Section* that a *Column* or a *Beam* includes is used to obtain cross-section dimensions.

- *IsJointConfined()* : Checks whether a *Joint* is confined or not (Section 2.3.4). The number of *Beams* connecting to a Joint is checked and "width" parameter of *Frame Cross Section* that *Beam* includes is compared with the "width" or "depth" parameters of *Frame Cross Section* that *Column* includes to check whether the *Joint* is confined or not.

- *ReturnMinAmountOfReinf_BCJ()* : Returns the minimum amount of transverse reinforcement at beam-column joints defined in TEC (Section 2.3.4.1).

- *ReturnMinTransReinfDia_BCJ()* : Returns the minimum transverse reinforcement diameter at beam-column joints defined in TEC (Section 2.3.4.1).

- *ReturnMaxTransReinfSpacing_BCJ()* : Returns the maximum transverse reinforcement spacing at beam-column joints defined in TEC (Section 2.3.4.1).

- *ReturnDesignShearForce()* : Returns the design shear force of a column defined in TEC (Section 2.3.3)

- *ReturnDesignShearForce_BCJ()* : Returns the design shear force of a beam-column joint defined in TEC (Section 2.3.4)

The methods of the *TurkishCodes Class* utilize the *TS500_2000* and *TEC Class* for performing all the predefined design and detailing tasks mentioned in Figure 4.47 :

- *CheckSectionDim()* : Compares cross-section dimensions of a column with the minimum cross-section dimensions specified by design codes. The "width" and "depth" parameters in *Frame Cross Section Class* (Figure 4.5) that *Column Class* includes (Figure 4.19) are set as current cross-section dimensions. These current dimensions are compared with the maximum value of the results of *ReturnMinSecDim()* methods of *TS500_2000 Class* and *TEC Class*.

- *CheckSectionArea()* : Compares cross-section area of a column with the minimum cross-section area specified by design codes. The "area" parameter in *Frame Cross Section Class* (Figure 4.5) that *Column Class* includes (Figure 4.19) are set as current cross-section area. This current area is compared with the maximum value of the results of *ReturnMinSecArea()* methods of *TS500_2000 Class* and *TEC Class*.

- *CheckReinfRatio()* : Compares longitudinal reinforcement ratio in a concrete column section with maximum and minimum longitudinal reinforcement ratios specified by design codes. *Column Class* includes *Reinforcement Pattern Class* (Figure 4.19). In *Reinforcement Pattern Class*, there exists *Longitudinal Pattern Class* (Figure 4.30). For each *LongitudinalBar* in the list *Longitudinal Pattern Class* keeps (Figure 4.42), the *Area()* method of *Longitudinal Bar* is called to calculate the area of a single bar and these calculated areas are summed for obtaining longitudinal reinforcement amount in a cross-section. Then, calculated longitudinal reinforcement amount is divided by the "area" parameter in *Frame Cross Section Class* (Figure 4.5) that *Column Class* includes (Figure 4.19) to obtain longitudinal reinforcement ratio. Finally, calculated longitudinal reinforcement ratio is compared with the minimum value of the results of *ReturnMaxLongReinfRatio()* methods of *TS500_2000 Class* and *TEC Class* and

maximum value of the results of *ReturnMinLongReinfDia()* methods of *TS500_2000 Class* and *TEC Class*.

- *CheckDia()* : Compares longitudinal and transverse reinforcement diameter in a column cross-section with maximum and minimum diameters specified by design codes. The "diameter" parameter of each *Longitudinal Bar* (Figure 4.31) in the list *Longitudinal Pattern Class* keeps (Figure 4.42), and each *Transverse Bar* (Figure 4.31) in the list *Transverse Patten Class* (Figure 4.45) keeps is compared with maximum value of the results of *ReturnMinLongReinfDia()* methods of *TS500_2000 Class* and *TEC Class* and maximum value of the results of *ReturnMinTransReinfDia()* methods *TS500_2000 Class* and *TEC Class*, respectively.

- *FrameType()* : Checks whether frames in a *Story Class* are sway or non-sway (Section 2.2.2). For each combination in the *Building Class*, total factored vertical load and horizontal story shear are computed from the element forces of the *Column Classes* in a *Story Class* (Figure 4.24) and relative lateral deflection of a *Story Class* is computed with the top and bottom story information obtained from the *Building Class*. The parameter ψ in Equation 2.8 is calculated with the computed force and drift values and compared with the limit value defined by specified design code. At the end of performing this procedure for both major directions, "isSwayX" and "isSwayY" parameters of a *Story Class* (Figure 4.24) are determined.

- *MomentMagnification()* : Calculates moment magnification factor for a column (Section 2.2.2.3). To decide whether a *Column* is sway or not, "isSwayX" and "isSwayY" parameters of a *Story* that the *Column* belongs to are used. Buckling load (N_k) is calculated with *Buckling Load()* method of Code Class. For each combination in the *Building Class*, factored vertical load (N_d), bending moments ($M_{d1}$and M_{d2}) are computed from the element forces of the *Column*.

- *DetEndStyle()* : Determines *End Style* parameters of each *Longitudinal Bar* in a Column for top and bottom joints of the column. For the bottom joint of the column, only "ver_main" parameter of *End Style* (Figure 4.41) that belongs to *Longitudinal Bar* (Figure 4.33) is determined. Firstly, *DevelopmentLength()* method of *TEC Class* is called for calculating the development length of *Longitudinal Bar* and then "ver_main" parameter of *End Style* is determined according to the length of the *Column*, development length and splice location of longitudinal bars. For the top joint of the column,

firstly, the existence of upper column at a *Joint* is searched in Columns connecting to the Joint. If the upper column exists, dimensions of the upper column are compared with the corresponding dimensions of the bottom column by utilizing "width" and "depth" parameters of *Frame Cross Section* stored in *Column*. If the corresponding dimension of the top column is equal to or smaller than the dimension of the bottom column, only "ver_main" parameter in *End Style* is calculated by utilizing the dimensions of beams connecting to the joint, length of the top column, development length and splice location. If the upper column does not exist at top joint, the parameters of *End Style* (Figure 4.41) are determined by utilizing "width" and "depth" parameters of *Frame Cross Section* stored in *Column* and *Beam*, and development length of a longitudinal bar.

- *DesignShearForce() :* Calculates the design shear force for a reinforced concrete column. For top and bottom joints of a column, *ChecksColumnsInAJoint()* method of *TEC Class* is called to understand whether the columns at a joint are stronger than beams at that joint. If the columns are stronger than beams, moment capacities of the columns are used to determine design shear force of a column (Section 2.3.3), otherwise moment capacities of the beams are used.

- *DesignShearForce_ BCJ()* : Calculates the design shear force for a beam-column joint. To calculate design shear force for a beam-column joint, the longitudinal reinforcement at the beam/beams connecting to the joint is used. Then, the calculated design shear force is compared with the limit design shear values determined according to type of the beam-column joint (confined or unconfined) (Section 2.3.4). To define the type of a beam-column joint as confined or not confined, *IsJointConfined()* method of *TEC Class* is called.

- *CheckReinf_ BCJ() :* Transverse reinforcement detailing such as diameter and spacing are kept in *TransversePattern Class* stored in a *Joint Class* (Figure 4.20). The diameter in *TransversePattern Class* is compared with the result of *ReturnMinTransReinfDia_ BCJ()* method of *TEC Class*. In addition to diameter, spacing in *TransversePattern Class* is compared with the result of *ReturnMaxTransReinfSpacing_ BCJ()* method of *TEC Class*.

- *SetLoadCombinations()* : Combines load cases to obtain most unfavorable loading conditions. *LoadCombination Classes* are developed according to specifications related

to loading combinations defined in *TS500_2000 Class* and stored in *Building Class*. These *LoadCombination* are utilized to obtain design forces in desiging and detailing of column and reinforcement.

CHAPTER 5

CASE STUDY

5.1 INTRODUCTION

The object library developed in this study was implemented and tested via a client-server based framework where multiple engineers can work on a single project simultaneously. The framework utilizes XML (Extensible Markup Language) web services technology. At the server side, all the design and detailing related information was kept as XML files and managed by the developed service. On the client side, a GUI was developed to communicate with the server, interact with the engineer while performing design and detailing calculations, and visualize the objects and results of the calculations. All the design related information is stored in XML format on the server side and no local copies of files are allowed on the client side.

First, the developed client GUI will be summarized and then the performance of proposed library will be tested by designing an actual six-story reinforced concrete building. The details of the server side services can be found at Anıl's master thesis [Anil, 2009].

5.2 CLIENT GUI

While designing and detailing a reinforced concrete structure, a GUI (Graphical User Interface) was developed in order to visualize the structure and to interact with engineers for performing design and detailing tasks. The GUI was developed with C#.NET programming language and OpenGL graphic library. The GUI works at the client side and communicates with the server for uploading and downloading design specific information. The general template of the GUI is presented in Figure 5.1.

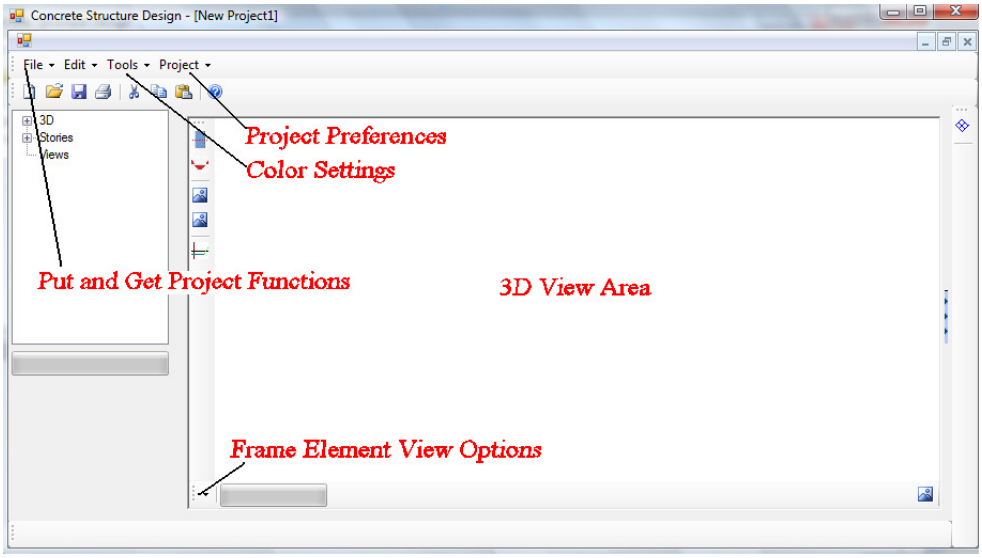

Figure 5.1: General view of the client GUI

The white area is the main part of the client GUI. In this part, 3D and 2D elevation and plan views are drawn. The user can rotate, pan or move the 3D views of the structure, visualize the columnLines and beamStrips and selects a columnLine for design and detailing.

The GUI was utilized not only for designing structural components but also for initialization of the design project. Currently, the framework does not have any modeling and analysis tools. Because of this reason, the structure is first modeled and analyzed by a third party analysis software and then the initial geometry and analysis results are imported to the server via the client GUI. For this purpose, "Put Structure" menu item of the GUI was utilized. While uploading the initial information of the structure to the server, it is the duty of the GUI to prepare the object definitions and XML files. "GetStructure" menu item of the GUI was utilized to retrieve the information about previously uploaded structure.

The GUI allows to determine some project parameters such as material classes, material models, avaible diameters of reinforcement with the "Project Preferences" menu. At the "Concrete" tab of the "Project Preferences" menu (Figure 5.2), concrete material class and concrete model that defines the stress-strain relationship of concrete can be defined for both columns and beams. On the other hand, steel material class and steel model that defines the stress-strain relationship of steel can be defined at the "Reinforcement" tab of the "Project Preferences" menu (Figure 5.3). Moreover, avaible diameters of reinforcement that will be used while designing can also be determined at this tab.

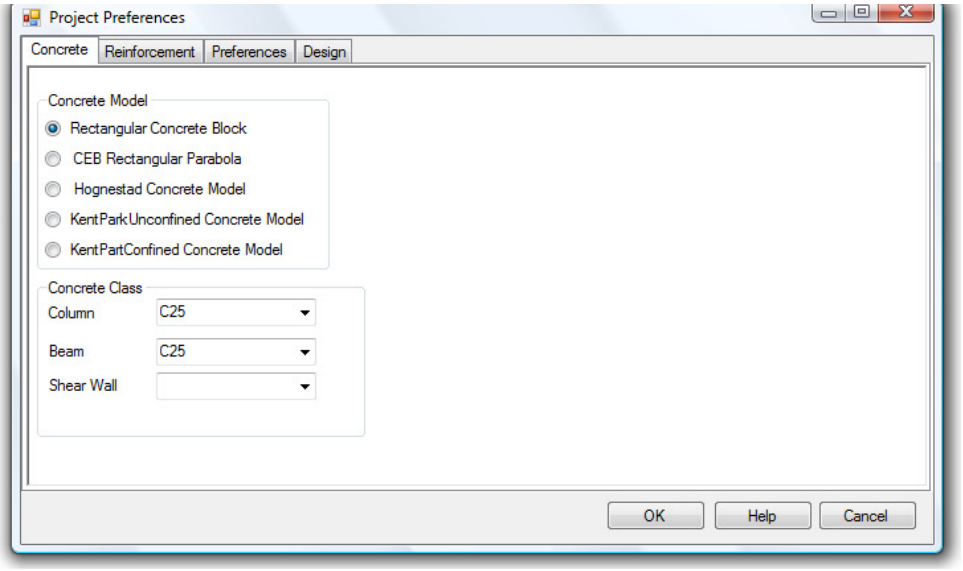

Figure 5.2: Concrete Tab of Project Preferences

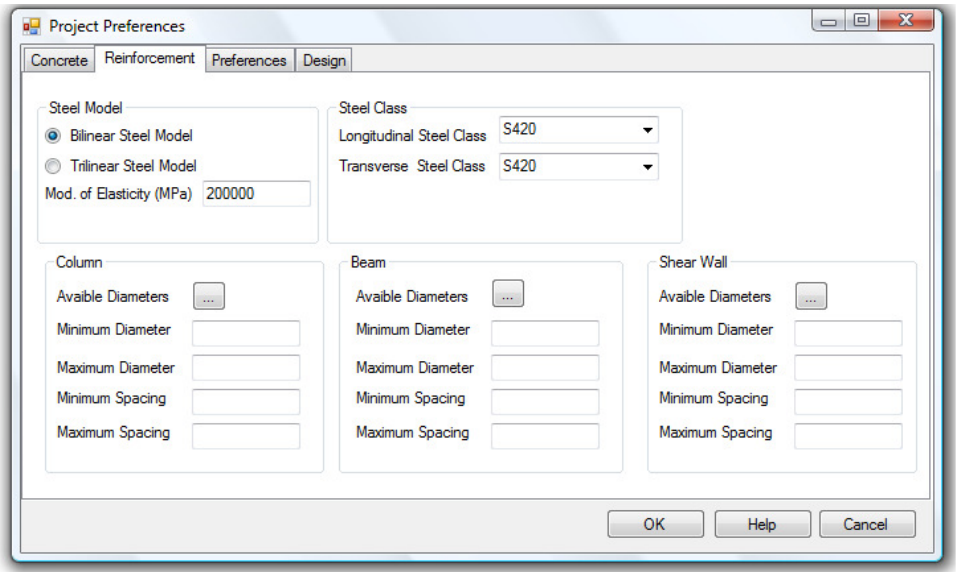

Figure 5.3: Reinforcement Tab of Project Preferences

While drawing the biaxial interaction diagram of a column cross-section, some parameters such as strip thickness which is the thickness of a slice when the compression zone of the section is divided into slices, maximum iteration, convergence value, and number of intermediate diagrams. The GUI allows to define these parameters at the "Preferences" tab of the "Project Preferences" menu (Figure 5.4). At the last tab of the "Project Preferences" menu (Figure 5.5), the design code is specified and load combinations are determined.

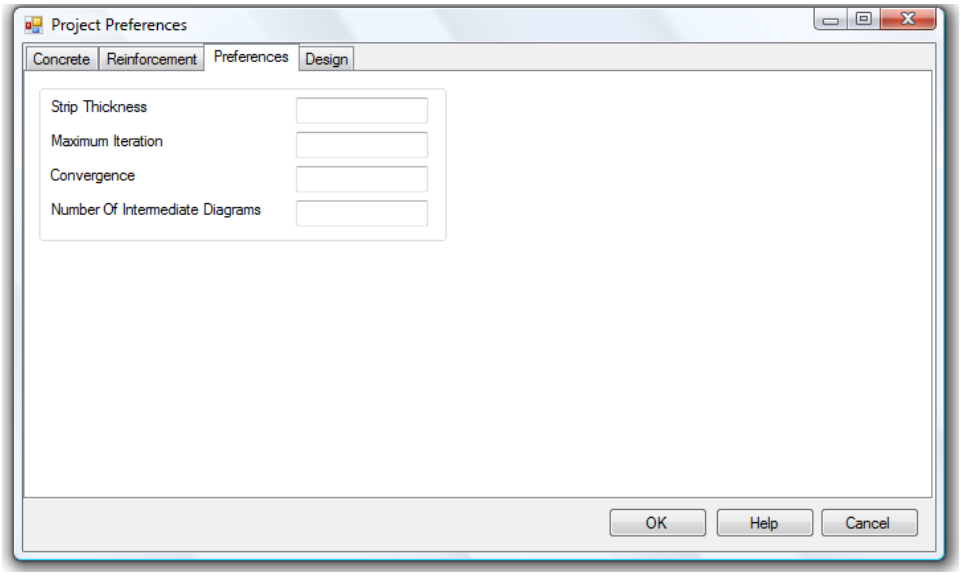

Figure 5.4: Preferences Tab of Project Preferences

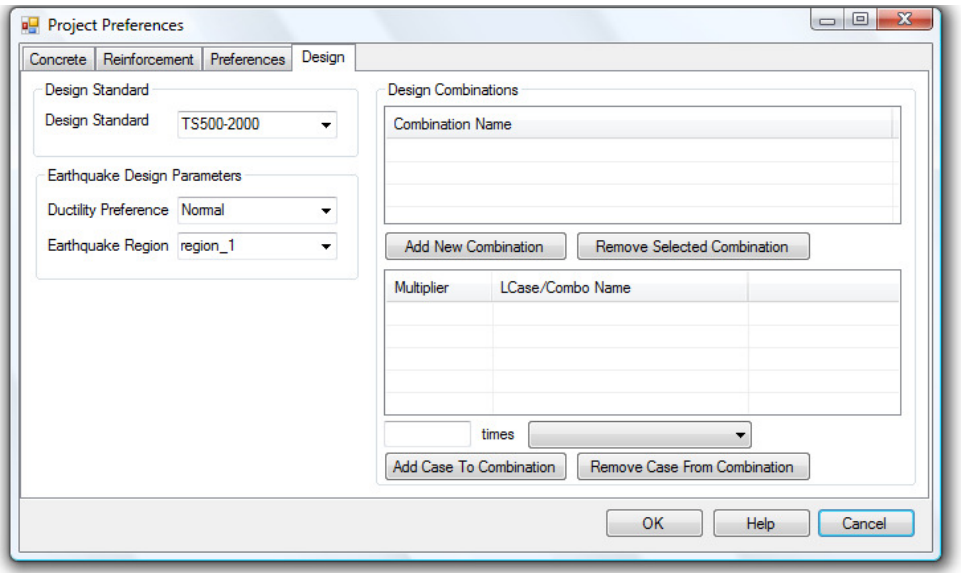

Figure 5.5: Design Tab of Project Preferences

5.3 IMPLEMENTATION

In order to test the performance of the proposed library, an actual six-story reinforced concrete building (Figure 5.6) was designed as a high ductility moment frame structure. As a starting point, the structure was first modeled and analyzed by ETABS v. 9.5, structural analysis and design software. Then, the analysis outputs such as analysis results at station points, cross-sectional properties, connectivity and coordinate information were exported to a Microsoft Access Database file by ETABS. Then, the created Microsoft Access Database

file was selected at the client GUI and classes belonging to both analysis and design entities packages were instantiated via the client GUI.

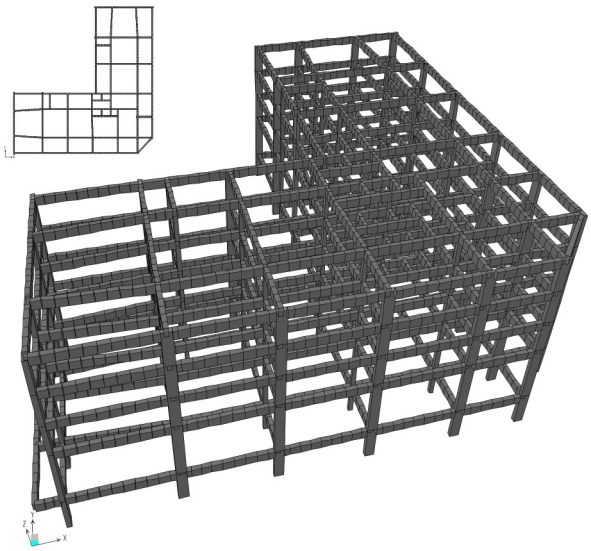

Figure 5.6: 3D view of the sample project

The main steps followed during the design and detailing of a ColumnLine are presented in Figure 5.7. The first step is the initialization step where the design parameters are determined, the initial geometry and analysis results are exported and all the objects belonging to analysis and design entities package are created. This step is performed once for each project by a single engineer. As the information of all objects are uploaded to the server, the second step, i.e, simultaneous design of ColumnLines is initiated. At this step, more than one engineer can download all the necessary object information for designing a single columnLine and by the help of the methods of Code object, the objects belonging to the reinforcement package are created. This way, the design of a columnLine is finalized. After that, engineers can prepare 3D CAD drawings for further checks of reinforcement placement.

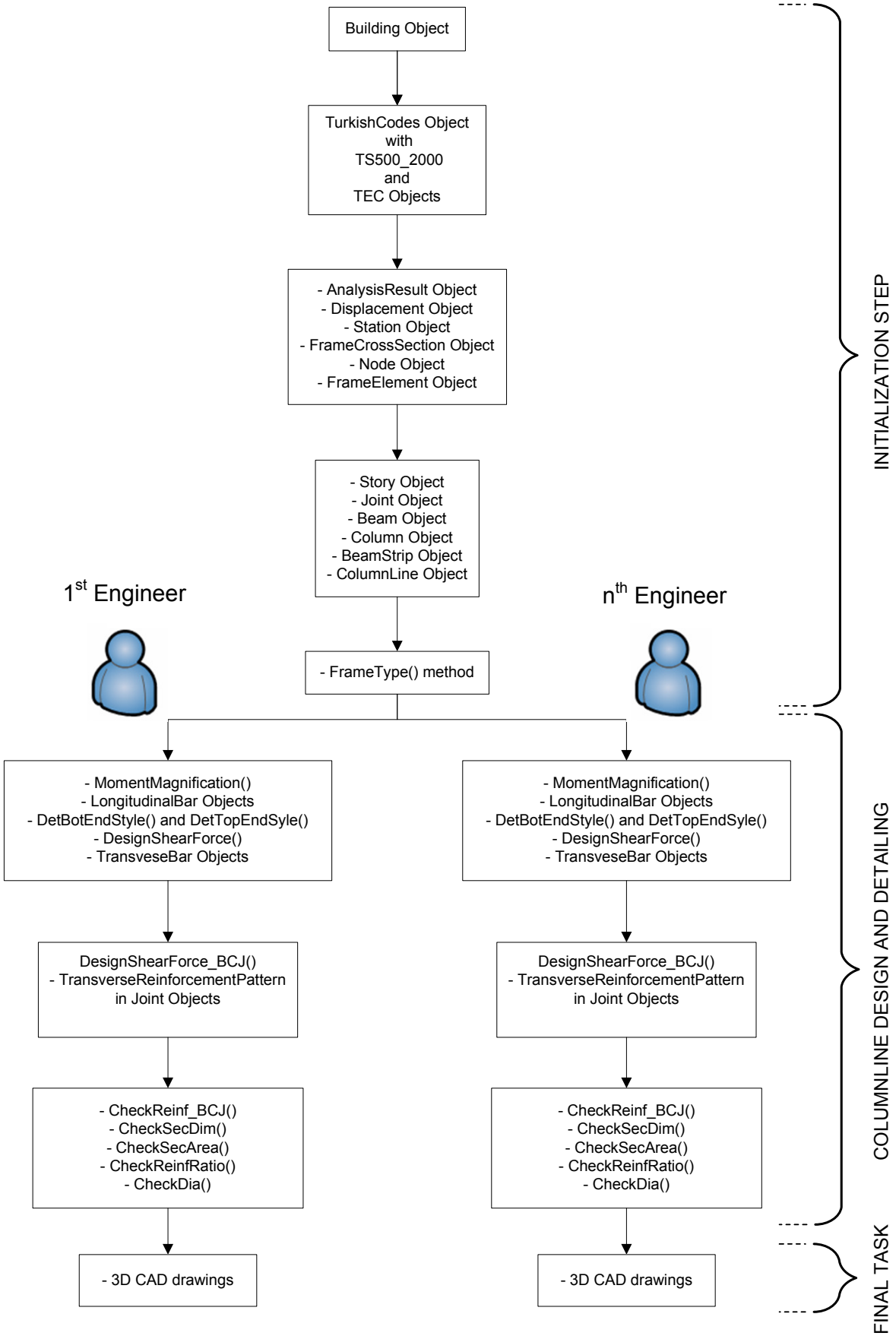

Figure 5.7: Flow Chart to Design a ColumnLine

5.3.1 Initialization Step

As a starting point, first the *Building Object* was initialized and the design parameters for *TurkishCodes Object* were determined. Next, objects belonging to the analysis package were created by utilizing the results of the analysis software. The created objects were *AnalysisResult, Displacement, Station*, *FrameCrossSection*, *Node* and *FrameElement Objects.* Then, higher level classes that utilize the classes in the analysis package were needed for design purposes. Therefore, *Story Objects were* firstly created and then *Joint, Beam* and *Column Objects* were created and stored in corresponding *Story Objects.*

Frame elements which are connected to a common node are stored in that *Node Class.* Moreover, each *FrameElement Object* keeps a parameter called "isPartOfAColumn" that indicates whether the frame element belongs to a column or not. These two information are used for grouping the frame elements of a story into columns by utilizing the algorithm presented in Figure 5.8. The algorithm basically goes over every frame element belonging to a story level and creates *Column Objects.* The algorithm is for classes where a column is modeled with more than one frame element. Otherwise, a column is usually composed of a single frame element.

Having detected columns at each story level, column lines were created with an algorithm similar to the one utilized for detecting the columns. The detected *ColumnLine Classes* were stored in *Building Class.*

The slenderness effects were considered by using by using the MomentMagnification method. Thus, after having created all design objects that define the geometry of a building, the *FrameType()* method of the code object was called for determining whether the frames of the stories are sway or not, and "isSwayX" and "isSwayY" parameters of each *Story Object* were determined. These parameters will be utilized later during the flexural design of each columnLine.

As all objects of the analysis and design entities package were create, they were uploaded to the server. At this step, the initial checks about the material properties and member sizes of the structure were performed utilizing the *CheckForce()*, *CheckSize()* and *CheckMaterialStrength()* methods of the *Code Object.* This step is performed once for each project.

5.3.2 Simultaneous Design of ColumnLines

After the initialization step, engineers can connect to the server and perform design and detailing tasks on *ColumnLine Objects* simultaneously. As a first step, only the model geometry that is composed of *FrameElement, Beam, Column, Joint, Story, BeamStrip* and *ColumnLine Objects* were downloaded from the server to each client GUI. To prevent unnecessary network usage, analysis results and existing reinforcement information were not downloaded at this step, yet. Figure 5.9 presents the 3D view of the structure at the client side.

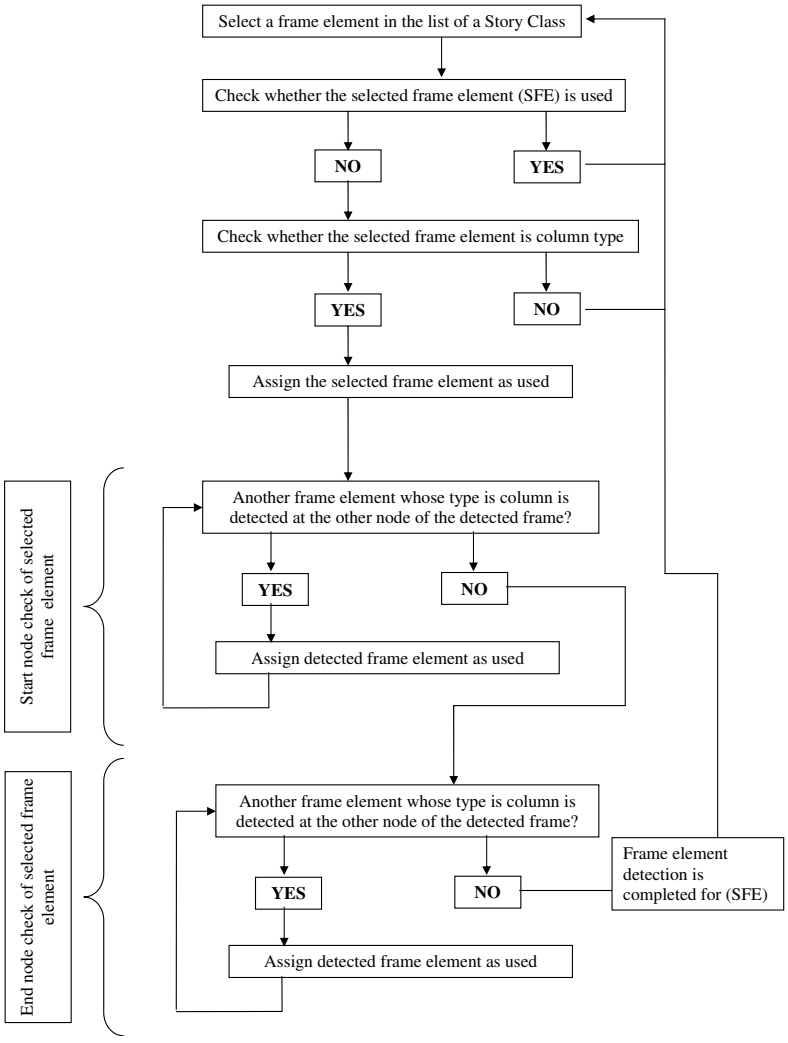

Figure 5.8: Pseudo Code for Detection of Frame Elements in the Same Column

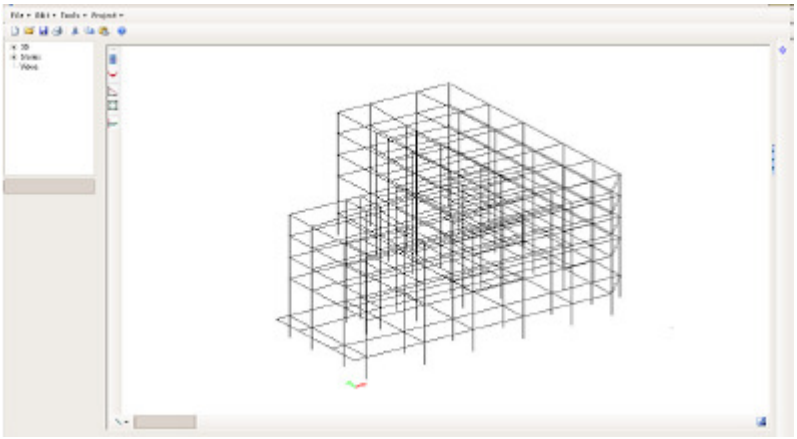

Figure 5.9: View of the sample project at client GUI

Design and detailing of each Column in a ColumnLine is performed simultaneously. Therefore, a column was selected to design and then properties of all other columns in the Column Line (Figure 5.10) that the selected columns belongs to were uploaded from the server side to the client side. On the other hand, in order to perform shear design of columns, the design of all beams connecting to columns must be finalized since beam's flexural capacities were required to compute design shear force. That's why when the column was selected to design, all properties of beams connecting to the Column Line that the selected column belongs to were uploaded, too. When the column is selected by the engineer at the client side, all columns in the Column Line and beams connecting to this Column Line are locked for modification and other engineers cannot modify this Column Line or connecting beams until the lock is released, but they can view the details.

The flexural design of columns were performed by utilizing the biaxial interaction diagrams at the *DesignSection Objects* located at the top and bottom ends of a column. Firstly, a predefined longitudinal bar pattern was chosen (Figure 5.11) at the client side and *LongitudinalPattern Object* was initialized by creating *LongitudinalBar Objects* with an initial diameter and positions determined according to selected predefined pattern. Then, biaxial interaction diagram was formed with the procedure defined in Section 3.4. For each load combination in *Building Object*, *MomentMagnification()* method of *TurkishCodes Class* was called and magnified design moments were obtained for each load combination. These moments including second order moment effects were checked according to formed biaxial interaction diagram. When the supplied longitudinal reinforcement was satisfied for all load combinations, the diameter of *LongitudinalBar Classes* was determined. Having defined the

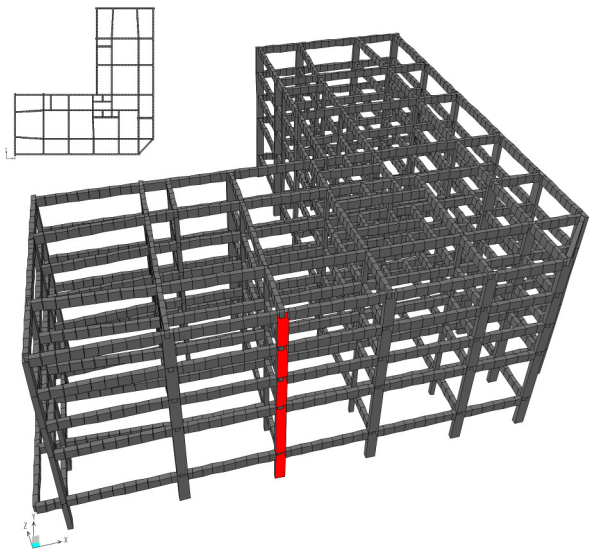

Figure 5.10: Selected ColumnLine in the sample project

diameter of *LongitudinalBar*, the end conditions of each *LongitudinalBar* were checked by *DetEndStyle()* method of *TurkishCodes Object* and *EndStyle Object* parameters for each *LongitudinalBar* were determined according geometric properties of the beam-column joint and calculated development length.

The shear design of columns was initiated by choosing a pre-defined transverse bar pattern (Figure 5.11). Then, the shear reinforcement areas for two orthogonal directions were calculated for the initial rebar diameter but the *TransversePattern* and *TransverseBar Objects* were created after having decided on the bar spacing for a specific DesignSection. The shear design was performed according to the design shear force calculated with the *DesignShearForce()* method of *TurkishCodes Object*. Three *DesignSections Objects* were defined per column for top and bottom confinement and mid zones. This way, different transverse reinforcement patterns and spacing were defined for each zone.

As the design of all columns in a ColumnLine was finalized, the beam-column joints were checked. For this purpose, the design shear force for all beam-column *Joints* in selected ColumnLine were calculated with the *DesignShearForce_BCJ()* method of *TurkishCodes Object* and transverse reinforcement amount and spacing were determined according to calculated design shear force. Then, transverse reinforcement detailing such as diameter and spacing are checked with the *CheckReinf_BCJ()* method of *TurkishCodes Object*.

As a final step, cross-section dimensions and cross-sectional area of each column were checked by *CheckSectionDim()* and *CheckSectionArea()* methods of *TurkishCodes Class*, respectively.

Then, the longitudinal and shear reinforcement ratios in the cross-section and reinforcement diameters used were checked by *CheckReinfRatio()* and *CheckDia()* methods of *TurkishCodes Class*, respectively.

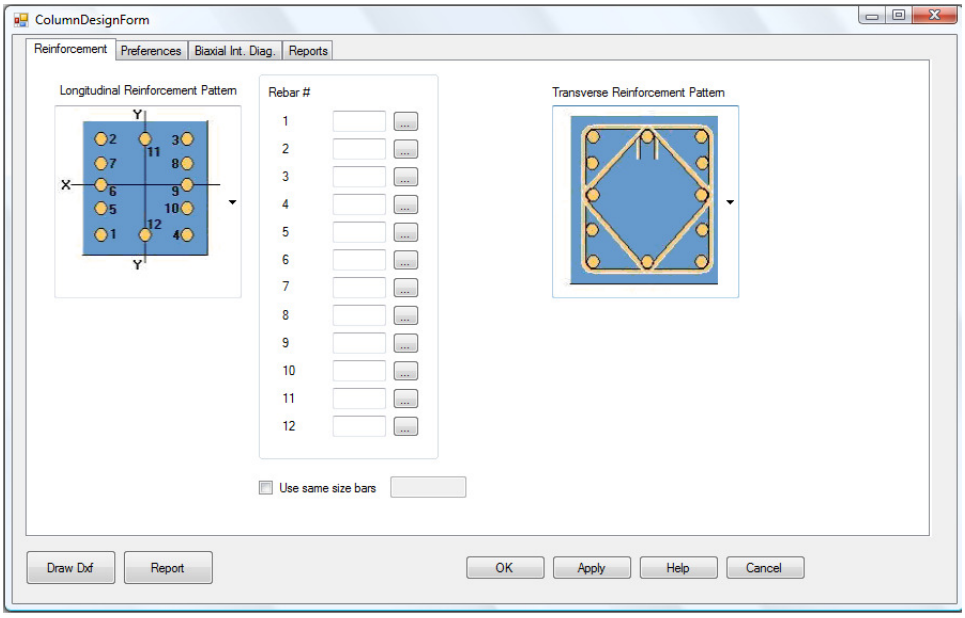

Figure 5.11: Selection of predefined patterns at client GUI

When the design and detailing of selected column line was finalized, the locks on selected column line and connecting beams to this column line are released. At this point, updated reinforcement information was uploaded to the server and other engineers were free to access to selected column line and connecting beams to selected column line.

As a final task, CAD drawings of the selected columnLine were prepared. Since the coordinates of all longitudinal and transverse bars in a column were stored with respect to the coordinates of the bottom joint of the column, there was no need to convert these coordinates to any other coordinate system. To draw reinforcement in a column, however, the coordinates of the bottom joint were added to the coordinates of the reinforcements of a column and the drawings were prepared. In these drawings, detailing of longitudinal reinforcement at the cross-section, splice location, transverse reinforcement spacing at beam-column joints, confined and unconfined regions of each column can be visualized. 3D CAD drawings of a selected column line in the sample project are illustrated in Figure 5.12, Figure 5.13 and Figure 5.14.

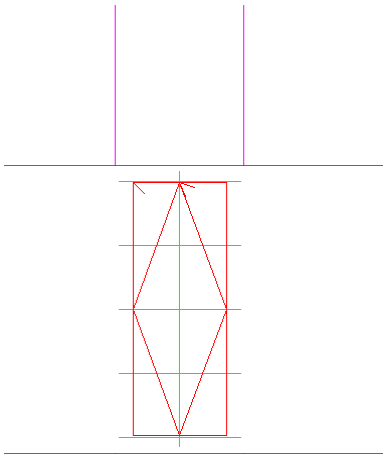

Figure 5.12: Section view of selected column line

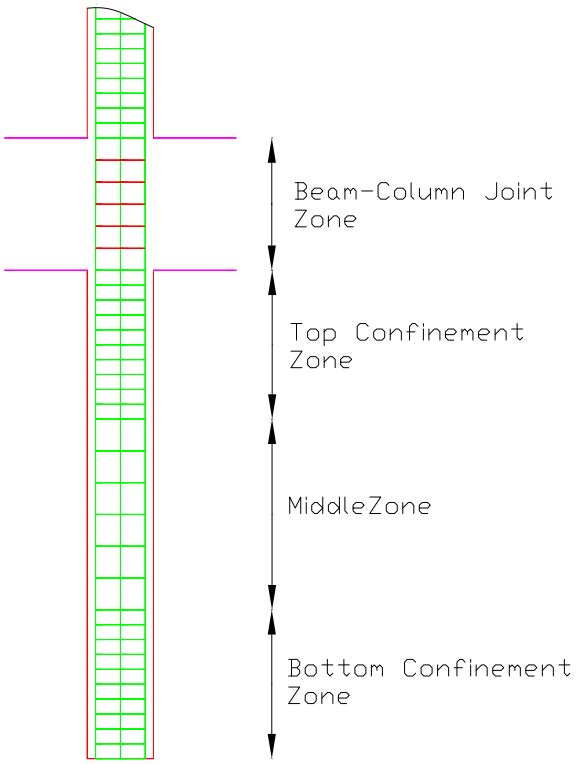

Figure 5.13: Side view of selected column line

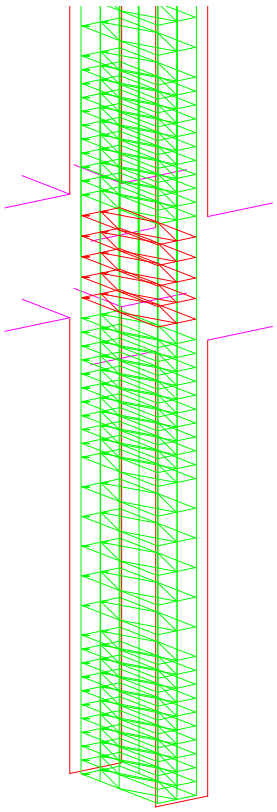

Figure 5.14: 3D view of selected column line

CHAPTER 6

CONCLUSION

While designing a reinforced concrete building, more than one engineer and draftsmen may be involved. Before and during the deign process, they share information. If synchronization between them is not performed well, the data to be shared may loss that will result in inadequate design and detailing of reinforced concrete buildings. Therefore, data management has a crucial role in design and detailing of reinforced concrete buildings.

In design and detailing of reinforced concrete columns, firstly, determined cross-sectional properties of columns are checked according to minimum code requirements. Then, the longitudinal and transverse reinforcement amount are calculated according to loading conditions and code requirements. In detailing of calculated reinforcement, longitudinal reinforcement mesh in a column cross-section, and placement of transverse reinforcement along the column height are determined according to a design code. Moreover, the transverse reinforcement at beam-column joints are detailed according to the size and load carrying capacities of beams and columns. Finally, shape and development length of longitudinal reinforcements are calculated according to geometry of the beam-column joints and design code and therefore splice locations of longitudinal reinforcement are determined.

To store all design and detailing information related to reinforced concrete columns, to perform code based design and detailing calculations and to minimize data loss between participant in a single project, a special data model is needed.

In this study, an object oriented data model storing dimensions of reinforced concrete columns, analysis results on these columns, detailed longitudinal and transverse reinforcement information along the column height, detailed shape and splice locations of longitudinal reinforcements, placement of transverse reinforcement at beam-column joints has been developed. One of the features of the developed data model is to perform code based calculations.

Moreover, the developed data model is implemented into a client-server environment that enables multi-user simultaneous design and detailing of reinforced concrete columns. Since the developed data model is centrally located at a server, data loss between participant in a single project is minimized.

In order to facilitate multi-user simultaneous design and detailing of reinforced concrete columns, a web based system has been implemented with a client - server relationship. For this relationship, XML Web Service Technology has been utilized. All information is kept in XML files and no calculation is performed at the server side. At the client side, however, design calculations are performed with developed GUI. With this GUI, it is possible to visualize reinforced concrete buildings in 3D view, to design and detail specified reinforced concrete columns, to get 3D CAD drawings of detailed reinforced concrete columns. When a column or column line is selected to design and detail by an engineer, it is locked and until the lock is released, other engineers cannot modify the selected column or column line.

The performance of the developed data model which was tested on an actual structure in Chapter 5 reveals that :

- The developed data model is capable of reflecting analysis results of the structure such as geometry of the structure, analysis results of the frame elements and displacements. To attain this, *FrameCrossSection, Displacement, AnalysisResults, Station, Node, FrameElement Classes* in the *Analysis Package* of the data model are utilized.

- *Story, Joint, Beam, Column, BeamStrip* and *ColumnLine Classes* are introduced in *Design Entities Package*. This way, not only the monolithic behavior of concrete structures is represented but also continuous columns on the same grid points are designed and detailed as a whole.

- *Code Class* in the data model enables the engineers to design and detail the reinforced concrete columns according to the specified design code. Moreover, having defined the main *Code Class* and determined overridden methods, it is easy to implement a new design code to the data model by just describing the content of these overridden methods according to the new code.

- The shape of a longitudinal bar at beam-column joints varies according to geometry of the beam-column joints, development length and splice location of the longitudinal bar. The complexity of having various shapes of longitudinal bars in beam-column joints are simplified by introducing *EndStyle Class*.

- Definition of the *Story Class* not only helps storing the Columns that are at the same level but also significantly simplifies the sway/non-sway frame definition.

- The definition of the *Joint Class* and its relationship with Column Objects allows the computation of joining Beam's flexural capacities thus simplifies the calculation of the design shear force for columns and beam-column joints.

- The definition of *MaterialModel Class* with *Concrete* and *Steel Class* enables the engineers to design and detail the members with different material strength and specify different concrete models for detailed section analysis.

- *DesignSection Class* facilitates the design at various locations of a column or beam. For example, transverse reinforcement details in top confinement zone, bottom confinement zone and mid zone of a reinforced concrete column are handled with three *DesignSecton Classes* stored in a *Column Class*. This way, different transverse reinforcement patterns and spacing can be defined at a single column.

- Having stored *FrameElement Objects* that connect to a common node in a *Node Class* facilitates to group frame elements that constitutes the same column with the node connectivity information. Similarly, to store Column Classes that connect to a common joint in a Joint Class facilitates to group columns at the same grid point with the joint connectivity information.

- Since the developed data model is implemented via a client-server environment. By allowing columnLines to be designed simultaneously, multiple engineers can work on a same project. The lock mechanism avoids the same components to be designed by more than one engineer. When a columnLine is locked by an engineer, other engineers cannot modify this column line, but they can work on another columnLine. This way, multiple design and detailing of reinforced concrete columns are performed.

- 3D drawings of a detailed *ColumnLine Objects* enable the engineers to visualize the reinforcement detailing such as confinement zone lengths, transverse reinforcement spacing along the column height, spacing at beam-column joints, longitudinal reinforcement arrangement in the column cross-section.

6.1 FUTURE RECOMMENDATIONS

In current data model, only the reinforced concrete columns and beam are designed and detailed. In the future, the number of reinforced concrete members to be designed and detailed can be increased with the design and detailing of slabs, shear walls, and foundations. Therefore, an engineer can design and detail a complete reinforced concrete building.

In this data model, analysis layer and design layer are separate from each other. In other words, analysis is performed by a different software program and design layer uses the results of the analysis layer such as structural geometry, column dimensions, column forces. In the future, analysis engine can be implemented to the developed data model. Therefore, a reinforced concrete building can be analyzed, design end detailed in a single framework.

REFERENCES

Anıl, Engin Burak, 2009 *A Web Based Multi-user Framework for the Design and Detailing of Reinforced Concrete Frames - Beams.* METU, ANKARA

Brunnermeier Martin , 1999 *Interoperability Cost Analysis of the U.S. Automotive Supply Chain.*

Bresler Boris *Design Criteria for Reinforced Columns under Axial Load and Biaxial Bending.*

Coble Richard J., Haupt Theo C.,Hinze Jimmie *The management of construction safety and health. 2000, Florida, USA*

DBYYHY, 2007. *Specifications for structures to be built in disaster areas.* General Directorate of Disaster Affairs, Turkish Ministry of Public Works and Settlement, ANKARA.

Deitel H. M. , Deitel P.J. , 2003. *Web Services - A Technical Introduction.* Pearson Education.

Eastman, Charles M., 1999. *Building Product Models: Computer Environments Supporting Design and Construction.* CRC Press.

Ersoy Ugur, Ozcebe Guney, Tankut Tuğrul 2004. Reinforced Concrete. Middle East Technical University, Ankara

Fowler Julian, *Step for Data Management, Exchange and Sharing.*

Gallaher, Michael P., O'Connor, Alan C., Dettbarn, Jr. John L., Gilday, Linda T., 2004. *Cost Analysis of Inadequate Interoperability in the U.S. Capital Facilities Industry.*

IAI, 2009. Ifc 2x edition 3 model definition. URL http://www.iai-international.org/Model/R2x3 final/index.htm.

International Alliance for Interoperability, *An Introduction to the International Alliance for Interoperability and the Industry Foundation Classes,* BETA - January 10, 1999

İlal Mustafa Emre, *The Quest for Integrated Design Systems : A Brief Survey of Past and Current Efforts.* 2007, METU JFA

Loffredo David , *Fundamentals of Step Implementation.* www.steptools.com/library

Peterson Dave, 2003. *The XML Schema Complete Reference.* Pearson Education.

Sfakianakis M.G , 2001. *Biaxial Bending with axial force of reinforced, composite and repaired concrete sections of arbitrary shape by fiber model and computer graphics.*

Stanek William R. , 2002. *XML.* Microsoft Press.W. Furlong Richard , Cheng-Tzu Thomas Hsu, and S. Ali Mirza, June 2004. *Analysis and Design of Concrete Columns for Biaxial Bending - Overview.* ACI Structural Journal

TS500, 2000. *Requirements for design and construction of reinforced concrete structures.* Turkish Standards Institute, ANKARA.

W. Furlong Richard , Cheng-Tzu Thomas Hsu, and S. Ali Mirza, June 2004. *Analysis and Design of Concrete Columns for Biaxial Bending - Overview.* ACI Structural Journal

APPENDIX A

CLASS RELATIONSHIP OF DATA MODEL

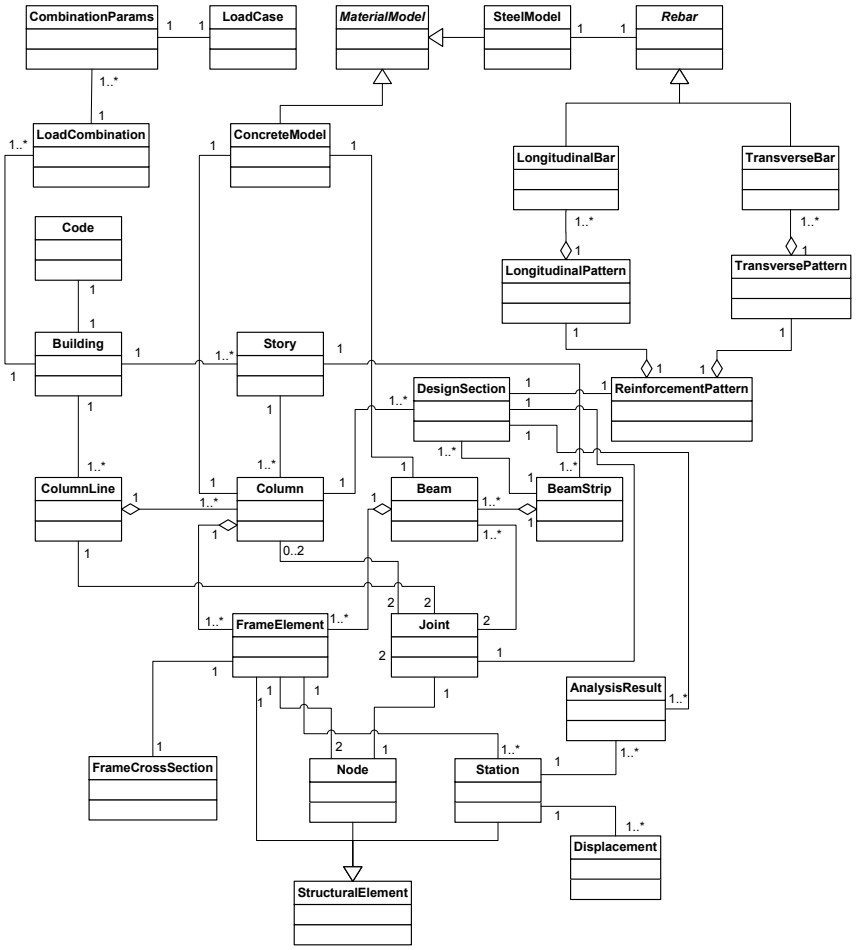

Figure A.1: Class Relationship of Data Model

APPENDIX B

COORDINATE TRANSFORMATION METHOD

Coordinate Transformation Method is used to recalculate the coordinates of a point when the coordinate system is rotated.

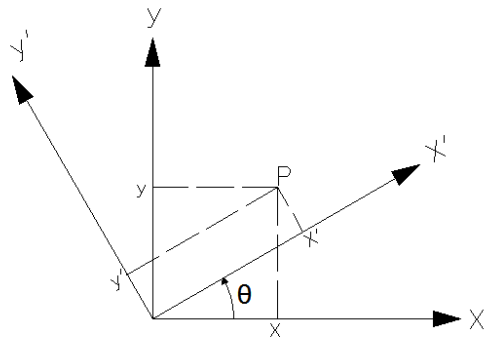

Figure B.1: Rotation of coordinate system

$$x' = x * cos\theta + y * sin\theta \tag{B.1}$$

$$y' = -x * sin\theta + y * cos\theta \tag{B.2}$$

To illustrate, x and y coordinates of point P in major axes are 40 and 60 respectively. When the coordinate system is rotated by 30^0 in counterclockwise direction, the coordinates ($x'$and y') in rotated coordinate system turn to

$$x' = 40 * cos30 + 60 * sin30 = 64.64$$

$$y' = -40 * sin30 + 60 * cos30 = 31.96$$