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

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

BY

ENGİN BURAK ANIL

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

JANUARY 2009

Approval of the thesis:

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

submitted by **ENGİN BURAK ANIL** in partial fulfillment of the requirements for the degree of
Master of Science in Civil Engineering Department, Middle East Technical University 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, **Civil Engineering**

Examining Committee Members:

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

Assoc. Prof. Dr. Uğur Polat
Department of Civil Engineering, METU _____

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

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

Dr. Semiha Kızıltaş
Department 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: ENGİN BURAK ANIL

Signature :

ABSTRACT

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

Anıl, Engin Burak

M.Sc., Department of Civil Engineering

Supervisor : Assist. Prof. Dr. Özgür Kurç

January 2009, 76 pages

Structural design of reinforced concrete structures requires involvement of many engineers who contribute to a single project. During the design process engineers have to exchange wide variety of information. Unmanaged data exchange may result in loss of resources. Developing a data model and setting up protocols for the management of data related to various structural design tasks can help to improve the quality of the structural design.

In this study, an object oriented data model was developed for reinforced concrete beams. Geometry of the structure, detailed shape and placement of the reinforcement, and design specific information for beams were defined in the data model. Design code based computations are facilitated by developing a code library.

Another focus of this study is developing 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 central management of design data. Design data was kept as XML files. Information was exchanged between the server and the engineer on a per-request basis. To design a beam strip, the engineer connects to the server

and chooses a series of connected beams. The strip that is selected is locked for modifications of other engineers to prevent any data loss and unnecessary duplicate efforts. When the engineer finalizes the design of a beam strip, data is updated on the server and the lock for this strip is released. Between these requests no active connection is required between the engineer and the server.

As a final task, the framework can produce structural CAD drawings in DXF format.

Keywords: multi-user, web based system, reinforced concrete frames, XML web services, object oriented data model

ÖZ

BETONARME ÇERÇEVELER İÇİN AĞ TABANLI TASARIM VE DETAYLANDIRMA ORTAMI - KİRİŞLER

Anıl, Engin Burak
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi : Yrd. Doç. Dr. Özgür Kurç

Ocak 2009, 76 sayfa

Betonarme binaların yapısal tasarımı birden fazla mühendisin tek bir projeye katkı sağlamasını gerektirir. Tasarım süreci boyunca yüksek miktarda ve çok çeşitli veri paylaşılır. Bir verinin gözden kaçması veya kaybolması kaynak ve zaman kaybına neden olabilir. Tasarım sürecinde ortaya çıkan veriyi yönetmek için bir veri modelinin yaratılması ve paylaşım kurallarının belirlenmesi tasarımın kalitesini arttırabilir.

Bu çalışmada, betonarme kirişler için nesne tabanlı bir veri modeli yaratılmıştır. Yapının geometrisi, donatıların şekli ve yerleşimi ve tasarım verileri veri yapısında tanımlanmıştır. Yönetmeliklere göre tasarım yapabilmek amacıyla bir yönetmelik kütüphanesi programlanmıştır.

Bu çalışmanın bir diğer amacı da, birden fazla mühendisin aynı anda bir betonarme bina üzerinde çalışmasını sağlayacak, ağ tabanlı bir ortamın oluşturulmasıdır. Veriyi merkezi olarak yöneten sunucunun programlanmasında XML Ağ Servisleri teknolojisi kullanılmıştır. Veri sunucuda XML biçiminde saklanmaktadır. Sunucu ve kullanıcılar arasında bilgi aktarımı istek doğrultusunda yapılmaktadır. Böylece ağ kaynaklarının gereksiz kullanımının engellenmektedir. Bir mühendis bir bina üzerinde çalışmak için sunucuya bağlanır ve tasarlayacağı kirişleri seçer. Seçilen kirişler diğer mühendislerin müdahalesini engellemek için kilitlenir.

Mühendis kirişlerin tasarımını bitirdiğinde kilit açılır ve tasarım bilgisi sunucuya gönderilir. Bu aktarımlar arasında işleyen bir ağ bağlantısına gerek yoktur.

Son bir aşama olarak tasarlanan kiriş detaylarının DXF biçiminde 3 boyutlu teknik çizimleri hazırlanmaktadır.

Anahtar Kelimeler: birden çok kullanıcı, ağ tabanlı sistem, betonarme çerçeve, XML ağ servisleri, nesne tabanlı veri yapısı

ACKNOWLEDGMENTS

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

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Computers and information systems have significant impacts on how business is conducted in almost every industry. Trading companies, automotive industry, post services, banks, and many others are taking advantage of new technologies and information based business concepts to make the business more efficient and profitable. Similarly, computer technologies are widely utilized in construction industry. Various disciplines contribute to a building project such as architects, structural engineers, and contractors. Each discipline needs special softwares and information to conduct their tasks. Large volume of information is produced through various tasks in a project. Hence, in order to effectively manage information in a building project, development, utilization and management of information systems must be specially addressed.

In construction industry today, many tasks related to the design process of building structures are being conducted with computers. Through the life cycle of a building, many kinds of computer systems are being employed. Survey engineers are utilizing various equipments to digitize geographical data. Architects and engineers are positioning the building on digital survey data with CAD systems. Structural engineers are designing the building to resist external effects with structural analysis and design softwares. Project management professionals are deciding on the execution plan with the aid of computers. These and any other unmentioned stages in a construction project are separate tasks carried out by different teams of professionals on different software systems. During all these stages a wide variety and a large amount of information is produced. Produced materials can be several CAD files, reports, and

various types of other computer files. Every software produces its own kind of file type for data storage. The process of construction projects require that all these files and documents can be used by other engineers and architects. Thus, softwares must be interoperable. Latest updates must also be available to all related participants whenever needed.

Engineers either use these computer files directly while performing their tasks or they re-input the information into their software systems. Generally, the second case occurs. For example, when two different teams are using different software packages for their tasks, generally these two software systems are not interoperable, since each software vendor uses its own file type for data storage. Then, engineers have to input the same information in each software in a different format which results in a duplicate effort. Integration of software systems is a still developing research area. According to an NIST report (Gallaher et al., 2004), the cost of inadequate interoperability to the U.S. construction industry was estimated to be $15.8 billion in the year 2002. Examples of inefficiencies mentioned in the report include manual reentry of data, duplication of business functions, and the continued reliance on paper-based information management systems. It is stated in the report that although the cost of inadequate interoperability constitutes between 1 and 2 percent of the annual industry revenue, small improvements in the efficiency potentially represent significant benefits.

Structural design of reinforced concrete frames is a separate design stage and requires complex data exchanges among its subtasks. Design of reinforced concrete frames initiates with the determination of the structural framing, initial member dimensions and material properties. If the design code that is applied requires certain limitations for member dimensions and material properties then, these requirements must be fulfilled. After the initial phase, frame is modeled for structural analysis. The internal forces and displacements obtained from the structural analysis are used in the design and detailing process. Through design code regulations and application of principles of mechanics, required reinforcement amount to withstand design forces are calculated. Based on the calculated amounts, reinforcement bar diameters are selected and bars are placed through the concrete members. Finally, the designed structure is checked for conformance to serviceability criteria of the design codes. At any phase of the design process, it may be realized that the requirements of the design codes can not be satisfied with the initial assumptions. Then the structural framing, and initial dimensions shall be revised, which requires re-analysis of the structural system. Hence, design and detailing of reinforced concrete frames is not a serial process but an iterative process of analysis, design,

and code checking. Design softwares can be integrated in a standardized manner through a common data model. Then, the data model must support calculation of reinforcement amount, performing detailing of reinforcing bars and design code based calculations.

In the traditional approach design information is transferred between the engineers through ftp sites, internet links, e-mails, and post services. In large scale projects, massive amount of information is produced and exchanged. A statistics concerning the amount of information produced and number of participants involved in a large scale project is presented in Eastman et al. (2008). In $10 Million or more project, 420 companies (including suppliers and sub-sub contractors) and 850 individuals may be involved. 50 different types of documents of a total of 56000 pages may be produced. Storage space required to hold this volume of information is estimated to be approximately 3 Giga bytes (Eastman et al., 2008). During the information transfers in such a project, it is highly probable that a document would be lost or an update would be missed. In such a case for example, a failure to update structural drawings with the latest design modifications may cause serious economical losses, inadequacy in structural safety or time delays. Management of information is not important in large scale projects only. Like many other stages of a building project, many engineers involve to contribute to a single project during the structural design. Each engineer designs a part of the structural system. As a consequence of the reinforced concrete design practice, design of a structural component affects the design of other components. Hence, engineers must be able to reach latest design information of structural components that are relevant to their calculations. Hence, information and engineering tasks between the participants must be coordinated, and simultaneous production and storage of design information must be managed. Through utilization of computer and network technologies, it is possible to bring engineers together in one computer environment to work on a single project simultaneously. If such an environment is produced and rules and protocols to support the management of information and engineering tasks are set up, information management problems can be resolved, latest updates could be made available to all participants as needed, and engineers could design a structure simultaneously regardless of their geographic location.

An information system that is capable of performing reinforced concrete design and detailing calculations in a web based environment has been developed in this study. A data model has been developed to define detailed geometry of reinforcing bars, geometry of the building, and aid in the calculations. Additionally, design code objects have been developed to perform

code based calculations. A client-server application has been developed to allow engineers to design and detail reinforced concrete beams simultaneously. The data model was utilized as the main data model of the system. Several algorithms have been developed to aid in the calculations.

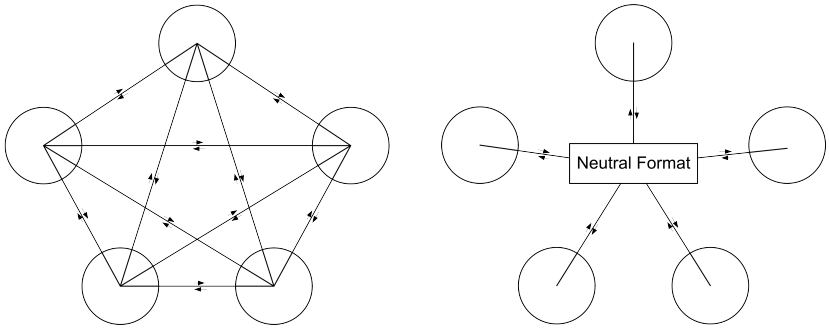
1.2 LITERATURE SURVEY

Selected studies from the literature concerning the development of information systems in the construction industry are briefly described below. As a major requirement, any information system should contain proper methods of describing the data stored or exchanged in the system. This can be achieved through using data models. Below, the data modeling studies in the construction industry are given first, then the web based systems are described.

1.2.1 DATA MODELS

In order to integrate softwares translator softwares or neutral file formats are developed. This way, information management and integration problem in the industry is aimed to be solved at the conceptual level (Bakis et al., 2007). Figure 1.1 presents the file exchange paths of translators and neutral file formats. The translator softwares translate file type of a particular software to file type of another software. The number of translators required to exchange data directly between n softwares equals $n(n-1)$. In this method, data is not exchanged directly between softwares but each software exports to and imports from the neutral file format. This method reduces the number of translators required to $2n$.

The requirement for integrating computer softwares was realized at the end of 1970's. Since then, there have been several standardization efforts at both national and international levels (Eastman, 1999). IGES, one of the first standard data model developed in the United States, aimed to define various data model and specifications for the description of geometrical data in CAD applications. At the IGES web site, the driving requirements and motivations are explained as follows : "In 1979, events took place that catalyzed the CAD vendor community to create the first national standard for CAD data exchange. Mechanical CAD systems were less than ten years old, and there were only a handful of products with any significant market penetration. Even at this early stage, users were overwhelmed by the inability to share data



(a) Direct exchange (b) Neutral file formats.

Figure 1.1: Exchange paths of direct exchange and neutral file formats.

among these tools and with their own internally-developed databases... General Electric (GE) challenged a panel of CAD vendors, that included ComputerVision, Applicon, and Gerber, to work together to enable a common neutral exchange mechanism..." (NIST, 1999). Similar efforts for developing standard data models took place in Germany for VDA-FS and in France for SET (Eastman, 1999).

Starting in the early 1980's, use of computer technologies in the construction industry increased (Eastman, 1999). Different software vendors started developing specialized softwares for separate engineering tasks. The intention in the programming world is that every software producer uses its own proprietary data model for storing data. As a result, at those days, a requirement for exchanging many different types of information emerged. Thus, exchanging geometrical information was not adequate any more. Other types of information from a broader engineering field should have to be exchanged. Since then, development of several product data models were initiated.

Among the earliest and largest efforts is the STEP (Standard for the Exchange of Product Model Data, ISO 10303), which was initiated by ISO (International Standards Organization) (TC184/SC4). STEP does not only covered construction but also many other fields of engineering such as automotive, aerospace, oil and gas, and process plants. In the construction industry, two international standards produced so far in STEP: *Building Elements Using Explicit Shape Representation* for exchange of 3D building models and *Plant Spatial Configuration* for exchange of spatial configuration of process plants. Also, there are other standards

under STEP that has direct or close relation to construction (Bakis et al., 2007).

Specifically aimed at the construction industry, GARM (General AEC Reference Model) and RATAS model can be mentioned from late 1980's (Eastman, 1999; Bjork, 1989). Similarly, CIMSteel were developed for "Computer Integrated Manufacture of Constructional Steelwork" (Watson and Crowley, 1994). CIMSteel is still in use today and is supported by several software vendors for product data exchange.

The latest international standardization effort in the construction industry is the development of Industry Foundation Classes (IFCs) (Bakis et al., 2007). The development of IFCs occurs under International Alliance for Interoperability (IAI). The latest IFC release (Release 2x Edition 3) covers several domains of construction industry such as structural analysis, architecture, HVAC (Heating, ventilation, and air conditioning), and facilities management (IAI, 2006). Japanese Chapter of IAI developed an extension in order to include reinforced concrete structural model to IFC (Yasaka et al., 2006). This extension covers definition of geometry of reinforcing bars in reinforced concrete members. IFC currently does not support design code based calculations and advanced detailing of reinforcement in reinforced concrete elements.

The construction industry is fragmented and characterized by adversarial behavior of different disciplines. Different disciplines have different views of the building components. For example, architects have a space-based understanding of a building while structural engineers have a component based view of frames, bays, etc. enclosing a space (Bakis et al., 2007). Hence, every discipline has different and special information needs. Additionally, during a project, transition in lifecycle phases leads to changes in objectives. Producing a complete neutral data format to satisfy the information needs of all disciplines is a very complex task and is not possible (Bakis et al., 2007). Instead of developing a complete neutral data format to reconcile all the information needs of all disciplines, special data formats can be utilized in the stages or tasks of a project, where appropriate. The data integration can be achieved through model evolution (Eastman and Jeng, 1999). Model evolution can exist in four modes: *i*) model translation where models are mapped into each other. *ii*) view generation where sub models of a model are generated. *iii*) modification where a model is incrementally modified by introducing new semantics. *iv*) evolution around a central model by mapping between multiple application views. The first type of model evolution is especially appropriate for stage transitions such as from design stage to the construction stage (Eastman and Jeng, 1999).

Translation of a model can be achieved by data mapping languages such as EXPRESS-X (Eastman, 1999) and XSLT (Clark, 1999) automatically. In data mapping languages rules are defined to translate a model into another. This method of translation has limited capability since it may not be possible to program the complete logic to map semantically different information (Bakis et al., 2007).

1.2.2 WEB BASED SYSTEMS

Several researches have been conducted to implement different network models and computer technologies to facilitate proper management of information through various stages of a construction.

Chassiakos and Sakellaropoulos (2008) implemented a web based system for managing construction information. The system consisted of a central server for managing and storing data and a web page to allow access to the data. The server side contained a database to store construction related information (construction equipment, activities, materials, etc.). A relational database management system (RDBMS) was employed to store such information. A data modeling procedure was conducted to determine the required tables, data fields, and relations between the tables in the database. The research adopted a data centric approach, opposed to a document centric approach. In the document centric approach, data is stored as documents (Microsoft Word files, CAD drawings, etc.). Documents are classified and identified with assigning external attributes to the documents (author, version, activity, date etc.). In the data centric approach, data is classified and identified with the properties and values defining the contents. As an example, storing the analysis results of a building in MS Word files and putting the files in a database exemplifies the document based approach. On the other hand, storing the results directly in a database as values of forces associated with structural members is the data centric approach.

Chen et al. (2006) implemented a web based system backed by a computer cluster for structural analysis computations. In the proposed system, users could connect to the system through a web browser and make structural analysis calculations on the server. The server side consisted of a PC cluster to perform computations and a management application to distribute the jobs to the computers in the cluster. This system provided a way of performing computations with minimal client side computational power.

Chen and Tien (2007) established a peer-to-peer network to facilitate CAD design over the network. In this approach, the users could form or join workgroups without utilizing a central server. In such workgroups, users can work on the same CAD drawing simultaneously. The CAD file was duplicated to all users in a workgroup to ensure that the latest modifications were always available to all users. Existence of duplicate copies results in the possession of the complete CAD file by every peer (user) at all times, instead of a piece of it.

Rosenman et al. (2007) developed a virtual environment to support multidisciplinary collaborative design. The different conceptual views and understanding of the same component by different disciplines are highlighted. Architectural and structural design is discussed as an example case. A view here is used in the sense that, every discipline has its own understanding of the problem domain. While, architects are more concerned with the spatial properties of a building, structural engineers are interested in load bearing properties of the structure. It is emphasized in the paper that for an environment to support multidisciplinary design, it must also support deducing multiple views from a set of data for each discipline. The information system contains a server for storing the information, and clients carry out their design tasks on the virtual environment presented.

Han et al. (1998) developed a client-server framework for online building code checking. This framework facilitated architectural code checking for disabled peoples needs. For information exchange, IFC was utilized. In this framework, IFC files produced in a compatible CAD package could be submitted through a web page to the server for code compliance checking.

Han et al. (1999) proposed a distributed architecture for building design services. The framework utilized CORBA (Pope, 1998) technology for the distributed system, and IFC for the product data model. CORBA is a network technology for building distributed systems. It is composed of a broker server for advertising the available services in the system, and several services for carrying out specific tasks. A new service first registers with the broker service to advertise itself in the system. A client application queries the broker to locate a specific service in the system. All exchange is channeled through the broker. The broker service acts as the meeting point of all services in the system. This way an application does not need to know where the service it needs resides (Han et al., 1999).

Similarly, Faraj et al. (2000) developed and implemented a computer environment to integrate design and construction. CORBA was utilized to build the environment. The environ-

ment supports CAD design, visualization, estimating, planning, specifications and supplier information. System enables users to share information through an IFC database.

Among other technologies utilized in the implementation of web-based systems, for the last several years, XML related technologies have found wide support in the programming world (Wolter, 2001). Relatively easy implementation, simple concepts, and platform independence have served the reputation of XML related technologies. Researchers in the construction industry have also realized the significance of XML (Bakis et al., 2007). Chen et al. implemented an Industry Foundation Classes (IFC) based web server for collaborative building design between architects and structural engineers where XML was used for data sharing mechanism. In this proposed model, a centered and shared database was developed to manage all related data. A project starts with the architect uploading an architectural design in the form of IFC to the server. Roles for structural engineers and architects are defined to be used in the environment. Based on these rules, the design advances through design iterations between the architect and the engineer, until a final design is obtained. The structural design task does not include detailed component design of the structure, only CAD design is concerned.

1.2.3 COLLABORATION TOOLS

There are several commercial tools aiming at providing project collaboration among participants. Features of a project collaboration product can include document management, role or task based collaboration, design collaboration, operations management, version management, file locking, and mark up. These products are available in several forms such as web-hosted, client-server, or self-hosted.

AutoCAD Buzzsaw (Autodesk, 2009) is a hosted service that supports managing project documents, design collaboration on CAD drawings, construction, bid and operations management. Buzzsaw has features such as file locking, version control, activity tracking, viewing and markup, and sending notifications to project members. Buzzsaw has a document centric approach rather than a model based approach.

Bentley ProjectWise (Bentley, 2009) is a document management tool. It provides content management, publishing, design review, and asset lifecycle management.

1.3 OBJECTIVES AND SCOPE

In this study, a web based framework has been proposed for designing and detailing reinforced concrete beams. In the proposed framework, a team of engineers can connect to the system and carry out engineering tasks on a structural system without geographical boundaries and without worrying about the information management issues.

In the context of the study, several object libraries were developed as a product data model to serve the special information needs of the reinforced concrete beam design and detailing.

Finally, the developed libraries were implemented into a web based system to facilitate the above mentioned features. The web based system is capable of managing the information that is stored in the server and allows multiple engineers to work on a single project simultaneously.

1.4 ORGANIZATION OF THE THESIS

This thesis is composed of six chapters and appendices. Introductory remarks and previous studies on developing information systems are given in Chapter 1.

Chapter 2 summarizes the general procedure of the design and detailing process of reinforced concrete beams, as guided by Turkish design codes. General code procedures and the computational background of this thesis are given in this chapter.

Chapter 3 explains the data model that is developed within the context of the study. The design philosophy applied in the development of the object libraries is described with reference to the common practice of reinforced concrete design

Chapter 4 explains the development of the web based system. The rules and protocols that have been set up for multi-user simultaneous work are described. General procedure of designing and detailing of a reinforced concrete frame in the developed system is explained.

Chapter 5 shows a sample demonstration of the system for an actual model.

Finally, Chapter 6 presents a summary and the conclusion of this thesis and presents future recommendations.

CHAPTER 2

DESIGN AND DETAILING OF REINFORCED CONCRETE BEAMS

In this chapter, general design and detailing procedure for reinforced concrete beams is presented with reference to the Turkish Earthquake Code (DBYYHY, 2007), and Turkish Reinforced Concrete Design Code (TS500, 2000). General requirements for design of reinforced concrete structures are explained in the Reinforced Concrete Code (TS500, 2000), and special requirements for the design of structures under earthquake loads are explained in the Earthquake Code (DBYYHY, 2007).

This study assumes that, all the procedures before reinforcement calculation is completed. The structure is modeled. Loads on the structure is determined and applied. The analysis is completed, and all the internal forces are computed. Starting from this point, general procedure will be briefly described in a procedural sequence. This chapter constitutes the computational background of the object libraries developed in this study.

2.1 FUNDAMENTAL PRINCIPLES AND ASSUMPTIONS

2.1.1 GENERAL

It is stated in the Reinforced Concrete Code (TS500, 2000) that, during the design, safety against collapse throughout the service life of the structure must be ensured. In addition to this, effects of cracking and deformations on the serviceability and strength of the structure must be limited.

Limit state design principles are adopted in the Reinforced Concrete Code. The loads are multiplied by partial factors and material strengths are divided by partial factors for further safety. Two kinds of limit states, *i*) ultimate limit state, and *ii*) serviceability limit state, must be considered during the design. Ultimate limit state requires that the structure withstands the design loads with an adequate factor of safety. Serviceability limit state requires that deformations and cracks on the structure are limited to a permissible level (Ersoy and Ozcebe, 2001).

Following are assumed for the design of reinforced concrete elements (TS500, 2000):

- Tensile strength of concrete is ignored,
- Perfect bonding between the reinforcing bars and the surrounding concrete is assumed to exist. Hence, strain of a reinforcing steel and the surrounding concrete layers are equal,
- Plane sections remain plane after deformation,
- When the ultimate capacity at a section is reached, the concrete strain at the outermost compression layer equals $\epsilon_{cu} = 0.003$,
- Stress-strain relation of steel is elasto-plastic, and is governed by the equation $\sigma_s = E_s\epsilon_s \leq f_{yd}$

2.1.2 MATERIAL PROPERTIES

Design strength of materials can be found by dividing the characteristic strength of materials by partial factors of safety. Partial factor of safety for concrete (γ_{mc}) is given as 1.5, whereas this value for steel (γ_{ms}) is 1.15. f_{ck} and f_{yk} are the characteristic compressive strength of concrete and yield strength of steel respectively.

Concrete :
$$f_{cd} = f_{ck}/\gamma_{mc} \tag{2.1}$$
$$f_{ctd} = f_{ctk}/\gamma_{mc} \tag{2.2}$$
Steel :
$$f_{yd} = f_{yk}/\gamma_{ms} \tag{2.3}$$

In the Reinforced Concrete Code (TS500, 2000), material properties are also defined. Modulus of elasticity (E_s) for all kinds of steel is given as 2×10^5 MPa, and ultimate strain of steel (ϵ_{su}) is given as 0.01.

Rectangular stress block assumption can be used for the modeling of compressive stress distribution of concrete at a section. The compressive stress value of the stress block can be taken equal to $0.85 f_{cd}$, where f_{cd} is the design compressive strength of concrete. Depth of the stress block is given as $k_1 c$. c here is the neutral axis depth. k_1 is the ratio of the equivalent stress block depth to the neutral axis depth. Values of k_1 varies between 0.85 and 0.70 according to the compressive strength of concrete.

Modulus of elasticity of concrete (E_{cj}) at j days of age is presented in Equation 2.4 (TS500, 2000). f_{cjk} denotes the characteristic compressive strength of concrete at j days of age. Shear modulus of concrete (G_{cj}) can be taken as 40 % of modulus of elasticity. Poisson ratio of concrete (μ_c) can be assumed as 0.2. Tensile strength of concrete is given with Equation 2.5.

$$E_{cj} = 3250 \sqrt{f_{cjk}} + 14000 \text{ MPa} \tag{2.4}$$

$$f_{ctk} = 0.35 \sqrt{f_{ck}} \text{ MPa} \tag{2.5}$$

2.2 INITIAL CHECKS

In order to design any building, design codes require that sizes of beams, strength of the materials, and maximum forces on the beams are checked for compliance.

2.2.1 SIZE CHECK

Width of beams (b_w) must be greater than 200 mm and less than the sum of the total height (h) of the beam and the column width (a). Height of the beams (h) must be greater than 300 mm and 3 times the slab height (t) (TS500, 2000). Notations are presented in Figure 2.1.

In the case of an earthquake resistant design, width of beams (b_w) must be greater than 250

mm and less than the sum of the total height (h) of the beam and the column width (a). Height of the beams (h) must be greater than 300 mm and 3 times the slab height (t). Height of beams cannot be greater than 3.5 times the beam width (b_w) (DBYYHY, 2007).

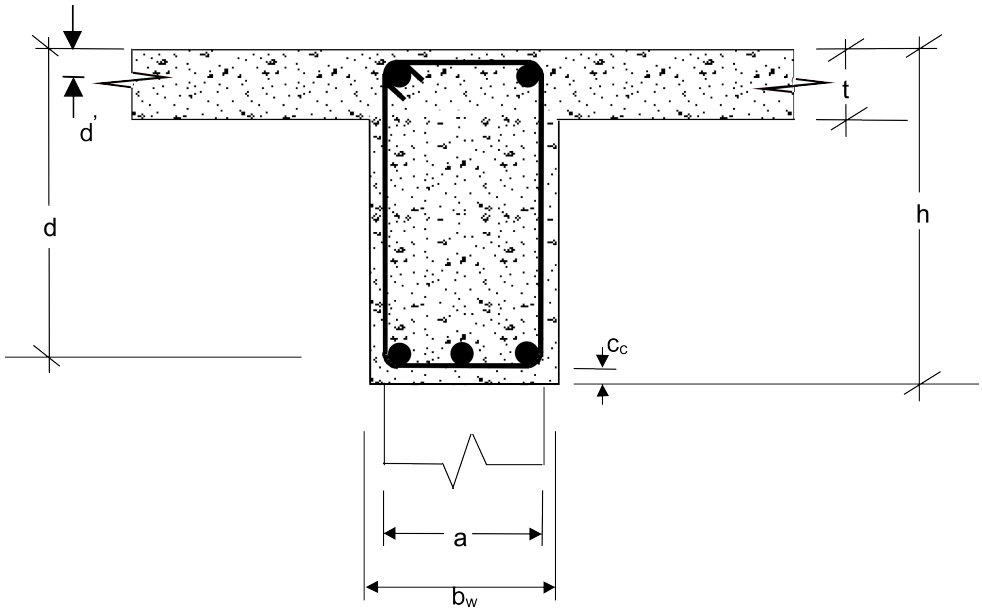


Figure 2.1: Beam size notations (TS500, 2000)

2.2.2 MATERIAL STRENGTH CHECK

TS500 (2000) requires that, the concrete strength (f_{ck}) should be greater than 16 MPa for structural members. If strength of concrete is greater than 50 MPa, TS500 cannot be used. Strength of reinforcement steel (f_{yk}) must be less than 420 MPa.

In earthquake resistant design, concrete strength (f_{ck}) must be higher than 20 MPa, and steel strength (f_{yk}) must be less than or equal to 420 MPa (DBYYHY, 2007).

2.2.3 FORCE LIMIT CHECK

Axial design force N_d on beams must be less than $0.1 f_{cd} A_c$, A_c being the cross section area (TS500, 2000; DBYYHY, 2007). Then, the element can be designed for pure flexure. If this condition is not satisfied, the element must be designed as a column for the combined effect

of bending and axial force.

Shear design force (V_d) on an element must be less than $V_{max} = 0.22 f_{cd} b_w d$. Otherwise, a larger beam size must be used (TS500, 2000).

2.3 DETERMINATION OF DESIGN FORCES

The codes require that, the design forces to be used for the reinforcement calculation in reinforced concrete frames are found with the superposition of different loading cases.

2.3.1 LOAD COMBINATIONS

In the design, all the following load combinations must be considered. In order to determine the design forces, TS500 (2000) requires the computation of the below load combinations as a minimum requirement for ultimate limit state design.

a. Gravity loads only,

$$F_d = 1.4G + 1.6Q \tag{2.6}$$

$$F_d = 1.4G + 1.2Q + 1.2T \tag{2.7}$$

In the above equations, G is the effects of dead loads, Q is the effects of live loads, and T is the effects of temperature change, shrinkage, differential settlement, and similar deformations and deflections.

b. Wind load combinations,

$$F_d = 1.0G + 1.3Q + 1.3W \tag{2.8}$$

$$F_d = 0.9G + 1.3W \tag{2.9}$$

W in the above equations is the effects of wind loads.

c. Earthquake load combinations,

$$F_d = 1.0G + 1.0Q + 1.0E \tag{2.10}$$

$$F_d = 0.9G + 1.0E \tag{2.11}$$

E in the above equations is the effect of earthquake loads.

d. Lateral earth pressure load combinations,

$$F_d = 1.4G + 1.6Q + 1.6H \tag{2.12}$$

$$F_d = 0.9G + 1.6H \tag{2.13}$$

H in the above equations is the effects of lateral earth pressure loads.

In the serviceability limit state design, factors of all loads in a combination are taken 1.0.

2.3.2 LOCATION OF DESIGN SECTIONS

According to the codes, values of design forces must be calculated at certain locations on the beams as a minimum requirement.

TS500 (2000) requires that bending moment (M_d) values are calculated at points of maximum value at spans, and at the column face around the supports for beams.

Values of shear force is calculated at a distance equal to the depth of the beam from the face of the columns. Maximum of the two values calculated at the two ends of a beam is used as the design shear force (V_d). In case of a beam-to-beam type of connection, shear value is calculated at the support face.

In the Earthquake Code (DBYYHY, 2007) moment values are calculated in a similar manner, but shear values are calculated at the faces of the columns, only.

2.3.3 VALUES OF DESIGN FORCES

According to Reinforced Concrete Design Code (TS500, 2000), flexural and shear design forces are obtained from a structural analysis for the load combinations explained in Section 2.3.1 at the locations described in Section 2.3.2.

In the Earthquake Code (DBYYHY, 2007), flexural design values are found similar to TS500 (2000). Shear design values are calculated with capacity design approach for high ductility beams. At the face of a column, the maximum shear load is reached when bending moment capacities are reached at the ends of the beam. The shear force in this condition is found by Equation 2.14. In this equation M_{pi} and M_{pj} are the plastic moment capacities at the ends of a beam, and V_{dy} is the shear force due to the gravitational loads. The plastic moment capacity at the beam ends can be found approximately by applying a factor of 1.4 to the elastic moment capacity (M_{ri}) (DBYYHY, 2007). Earthquake forces must be considered in both directions and the maximum value calculated by Equation 2.14 must be used in the design. For normal ductility beams, value of the shear design force is calculated as defined in TS500 (2000).

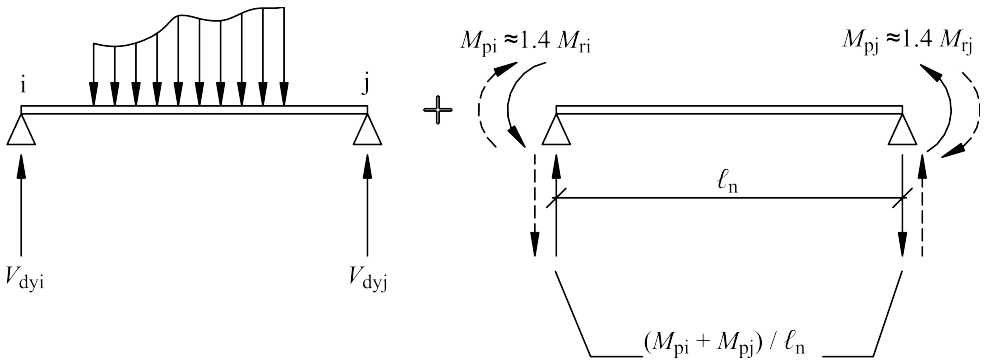


Figure 2.2: Calculation of shear design force (DBYYHY, 2007)

$$V_e = V_{dy} \pm \frac{M_{pi} + M_{pj}}{l_n} \tag{2.14}$$

2.4 DESIGN CALCULATIONS

After the design force values are calculated, the amount of reinforcement must be calculated.

2.4.1 FLEXURAL DESIGN

One way of calculating moment capacity of a beam section is presented in Equation 2.15 (Ersoy and Ozcebe, 2001), where *jd* is the length of the lever arm of the reinforcing bars.

$$M_r = A_{st} f_{yd} jd \tag{2.15}$$

j can be calculated by Equation 2.16. A_{st} is the tensile reinforcement area in a section. f_{yd} is the yield strength of steel bars.

$$j = 1 - \frac{k_1}{2}\frac{c}{d} = 1 - 0.59\rho\frac{f_{yd}}{f_{cd}} \tag{2.16}$$

$$\frac{k_1 c}{d} = \frac{A_s}{b_w d}\frac{f_{yd}}{0.85 f_{cd}} = \rho\frac{f_{yd}}{0.85 f_{cd}} \tag{2.17}$$

These equations are valid only for single reinforced concrete underreinforced sections, where steel has already yielded when strain at the outermost concrete layer in compression has reached $\epsilon_{cu} = 0.003$. Otherwise, compatibility equations must be used.

If Equation 2.18 is satisfied then the section is considered underreinforced. Underreinforced reinforcement ratios prevent brittle failure therefore, Turkish codes only permit underreinforced sections (TS500, 2000; DBYYHY, 2007).

$$\rho = \frac{A_s}{b_w d} \leq \rho_b = \frac{0.85 f_{cd}}{f_{yd}} k_1 \frac{0.003 E_s}{0.003 E_s + f_{yd}} \tag{2.18}$$

If a single layer of reinforcement cannot provide adequate moment capacity within the reinforcement amount limitations at a section, then either the size of the section should be enlarged or additional compressive bars should be placed at the compression zone. Moment capacity of a double reinforced section can be calculated as shown in Figure 2.3, where contribution of concrete and reinforcement can be divided into two force couples. Sum of the moment of these two force couples yields the moment capacity of the double reinforced section.

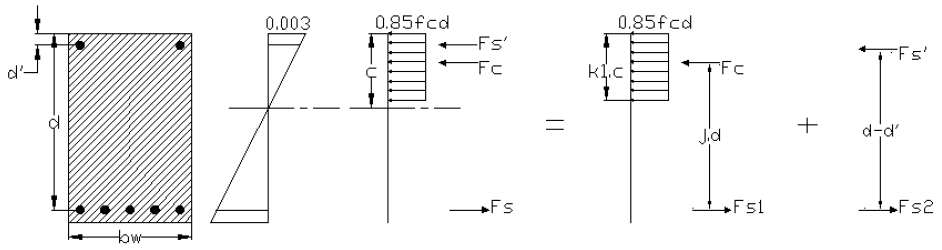


Figure 2.3: Calculation of flexural capacity of a rectangular section (Ersoy and Ozcebe, 2001)

An alternative procedure for designing a rectangular beam section is presented in Ersoy and Ozcebe (2001), and is as following:

a. Calculate K.

$$K = \frac{b_w d^2}{M_d} = \frac{1}{\rho f_{yd} j} \tag{2.19}$$

K does not have a physical meaning. K is useful for limiting the deflections during the design.

b. Calculate K_l.

$$K_l = \frac{1}{\rho_l f_{yd} j_l} \tag{2.20}$$

$$\rho_l = 0.235 \frac{f_{cd}}{f_{yd}} \tag{2.21}$$

$$\frac{c_l}{d} = \frac{0.276}{k_1} \tag{2.22}$$

$$j_l = 0.86 \tag{2.23}$$

When design moment is limited to a value that K of the section is less than K_l, excessive deflections of a beam can be prevented. If K is greater than K_l, K can still be checked against a maximum value K_m (Equation 2.24). In this case, deflections on the beam should be checked. K_m limits the amount of reinforcement that can be placed in a section, thus limiting the maximum moment value. It is advisable that K is limited to K_l.

$$K_m = \frac{b_w d^2}{M_m} = \frac{1}{\rho_m j_m f_{yd}} \tag{2.24}$$

ρ_m is the maximum reinforcement ratio permitted by the design codes. In Turkish design codes, ρ_m is given as $0.85\rho_b$. c_m is the depth of the neutral axis when capacity of a section with maximum allowed reinforcement ratio is reached. j_m is the ratio of the lever arm of the tension steel area to the section depth when capacity of a section with maximum allowed reinforcement ratio is reached. c_m and j_m can be calculated by applying principles of mechanics and reinforced concrete.

c. If $K \geq K_l$, section size is OK. Calculate tension steel area,

$$A_{st} = \frac{M_d}{f_{yd} j d} \tag{2.25}$$

d. If $K < K_l$, compression bars should be used. In this case, equate the first force couple to M_l as presented in Figure 2.3.

$$M_1 = M_l = \frac{b_w d^2}{K_l} \tag{2.26}$$

Moment of the second force couple,

$$M_2 = M_d - M_1 \tag{2.27}$$

$$A_{s1} = \frac{M_1}{f_{yd} j_l d} \tag{2.28}$$

e. Check if the compression bars will reach the yield strain. Strain of the compression bars can be calculated with Equation 2.29.

$$\epsilon'_s = \frac{\frac{0.276}{k_1} - \frac{d'}{d}}{\frac{0.276}{k_1}} \times 0.003 \tag{2.29}$$

f. If $\epsilon'_s \geq \epsilon_{sy}$, then compression bars will yield and stress on the compression bars will reach the yield stress, $\sigma'_s = f_{yd}$. Calculate the area of compression bars with Equation 2.30.

$$A_{s2} = A'_s = \frac{M_2}{f_{yd}(d - d')} \tag{2.30}$$

Total tension steel area,

$$A_s = A_{s1} + A_{s2} \tag{2.31}$$

g. If $\epsilon'_s < \epsilon_{sy}$, compression bars will not yield. Stress on compression bars are governed with Equation 2.32.

$$\sigma'_s = \epsilon'_s E_s \tag{2.32}$$

$$A_{s2} = \frac{M_2}{\sigma'_s(d' - d)} \tag{2.33}$$

Equate the components of the second force couple, and solve for the area of the compression bars A'_s.

$$A_{s2} f_{yd} = A'_s \sigma'_s \tag{2.34}$$

Compressive bar area,

$$A'_s = \frac{f_{yd}}{\sigma'_s} A_{s2} \tag{2.35}$$

Total tension reinforcement area,

$$A_s = A'_s + A_{s2} \tag{2.36}$$

Limitations for reinforcement amount in a section are as following:

$$\rho = \frac{A_{sw}}{b_w d} \geq \rho_{min} = 0.8 \frac{f_{ctd}}{f_{yd}} \tag{2.37}$$

$$\rho \leq 0.02 \tag{2.38}$$

$$\rho - \rho' \leq 0.85 \rho_b \tag{2.39}$$

2.4.2 SHEAR DESIGN

A procedure for providing capacity against shear forces is given below (Ersoy and Ozcebe, 2001):

a. Calculate shear design force, V_d,

b. Calculate cracking strength,

$$V_{cr} = 0.65 f_{ctd} b_w d \psi \tag{2.40}$$

If axial load can be ignored, then ψ can be taken as 1.0. Otherwise,

$$\psi = 1 - 0.3 \frac{N_d}{A_c} \quad \text{, (If } N_d \text{ is compression)} \tag{2.41}$$

$$\psi = 1 + 0.07 \frac{N_d}{A_c} \quad \text{, (If } N_d \text{ is tension)} \tag{2.42}$$

c. If $V_d \leq V_{cr}$, there is no need to calculate shear reinforcement thus, providing the minimum amount of reinforcement is adequate.

$$\min \rho_w = \min \frac{A_{sw}}{s b_w} = 0.3 \frac{f_{ctd}}{f_{ywd}} \tag{2.43}$$

d. If $V_{max} = 0.22 f_{cd} b_w d \geq V_d > V_c$, shear reinforcement must be calculated. Definition of V_c is given in the next paragraph.

$$\frac{A_{sw}}{s} = \frac{V_d - V_c}{f_{ywd} d} \geq \min(\rho_w) b_w \tag{2.44}$$

V_c is the contribution of concrete to the shear capacity of a section. In design, V_c is specified as $0.8 V_{cr}$ in TS500 (2000). In earthquake resistant design, according to Section 3.4.5.3 of the Earthquake Code (DBYYHY, 2007), at the confinement regions of the beams, if shear force due to the earthquake forces are greater than 50% of the total design shear force (V_e), then the contribution of concrete to the shear capacity V_c must be ignored. Confinement regions are those that cover a length of $2h$ from the support face on both ends of the beams (DBYYHY, 2007).

2.5 DETAILING OF REINFORCEMENT

Detailing rules of longitudinal and transverse bars are explained in this section.

2.5.1 LONGITUDINAL BARS

2.5.1.1 PLACEMENT

According to TS500 (2000), the size of longitudinal bars for flexure must be greater than ϕ 12. In a section, distance between two bars must be larger than either 20 mm, bar diameter or $4/3$ of the largest aggregate size.

2.5.1.2 EXTENSION, DEVELOPMENT, AND SPLICING

According to TS500 (2000), at least $1/3$ of the tensile longitudinal reinforcement in the span regions of beams must be extended and developed into the supports.

Reinforcing bars must be adequately developed into the concrete in both directions in order to carry the design forces (TS500, 2000). Providing adequate amount of adherence between concrete and the steel bars depends on the location of reinforcing bars during the pouring of concrete.

CASE I Bars that are not in CASE II.

CASE II Bars that have an angle between 45^o and 90^o with the horizontal axis. Bars that have an angle less than 45^o with the horizontal axis and that are in the lower half of the section or that are at least 300 mm away from the top surface of the beam section.

Minimum development length of tension bars should be calculated according to Equation 2.45. If the bar diameter is between *32* mm and *40* mm, l_b must be increased by a factor of $100/(132 - \phi)$. Development length calculated by Equation 2.45 must be increased by 40% for bars that are in CASE I.

$$l_b = 0.12 \frac{f_{yd}}{f_{ctd}} \phi \geq 20\phi \tag{2.45}$$

If bar ends are hooked, development length can be reduced. Hook types are shown in Figure 2.4. Straight part of the hooked bar l_{bk} can be reduced to 3/4 of the value calculated by Equation 2.45.

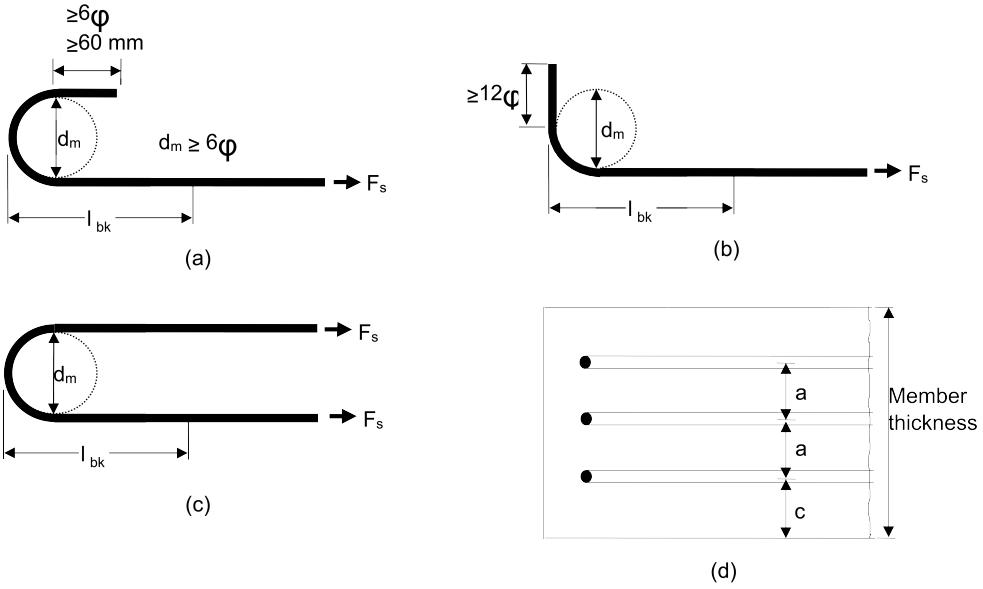


Figure 2.4: Standard hooking types (TS500, 2000)

Compression bars in a section cannot be hooked. Minimum development length for compression bars can be taken as $3/4 l_b$ where l_b is calculated with Equation 2.45.

Bars can be spliced as allowed by the design codes. In this thesis, only lap splices are considered. Lap splicing rules of reinforcing bars is discussed in the following paragraphs.

Minimum length of lap splices for tension bars is calculated with Equation 2.46. r in the Equation 2.47 is the ratio of the lap spliced reinforcement area to the total reinforcement amount at a section. If all the reinforcing bars are in tension, then value of α_1 is taken to be 1.8. For bars that are considered in CASE II, l_0 should be multiplied by a factor of 1.4. If spliced bars are hooked, l_0 can be reduced by 1/4. Along the splice, confining transverse

reinforcement must be placed. Diameter of the confining transverse reinforcement must be greater than 1/3 of the spliced bar diameter or ϕ 8. Along the splice, at least 6 confining transverse bars must be placed, and transverse bar spacing must be less than $1/4d$ and 200 mm. If, more than one bar is spliced in a section, distance between the centers of the splices must be greater than $1.5l_0$.

Length of lap splices for bars in compression must be greater than l_b calculated with Equation 2.46 or 300 mm. Spacing of confining transverse reinforcement around the lap splice of bars in compression must be less than $d/4$. Bars having a diameter larger than 30ϕ cannot be lap spliced.

$$l_0 = \alpha_1 l_b \tag{2.46}$$

$$\alpha_1 = 1 + 0.5r \tag{2.47}$$

Additional requirements for earthquake resistant design are described in the following paragraphs emphasizing the Turkish Earthquake Code (DBYYHY, 2007). In earthquake resistant design of beams, at least two longitudinal bars must be placed continuously through the span of the beams at the top and bottom of the beam sections. Amount of longitudinal reinforcement at a section must follow the rules in TS500 (2000).

In 1^{st} and 2^{nd} earthquake regions, at the supports of the beams, the amount of bottom bars cannot be less than 50% of the amount of top bars at the same section. In 3^{rd} and 4^{th} earthquake regions, at the supports of the beams, the amount of bottom bars cannot be less than 30% of the amount of top bars at the same section.

1/4 of the larger of the amount of top bars at the two support sections of a beam must be placed continuously through the beams. Rest of the top bars at the support sections must be placed according to TS500 (2000).

In case a beam is not continuous, top and bottom bars at the support sections must be extended to the other side of the confined core of the column and bended 90^o into the transverse

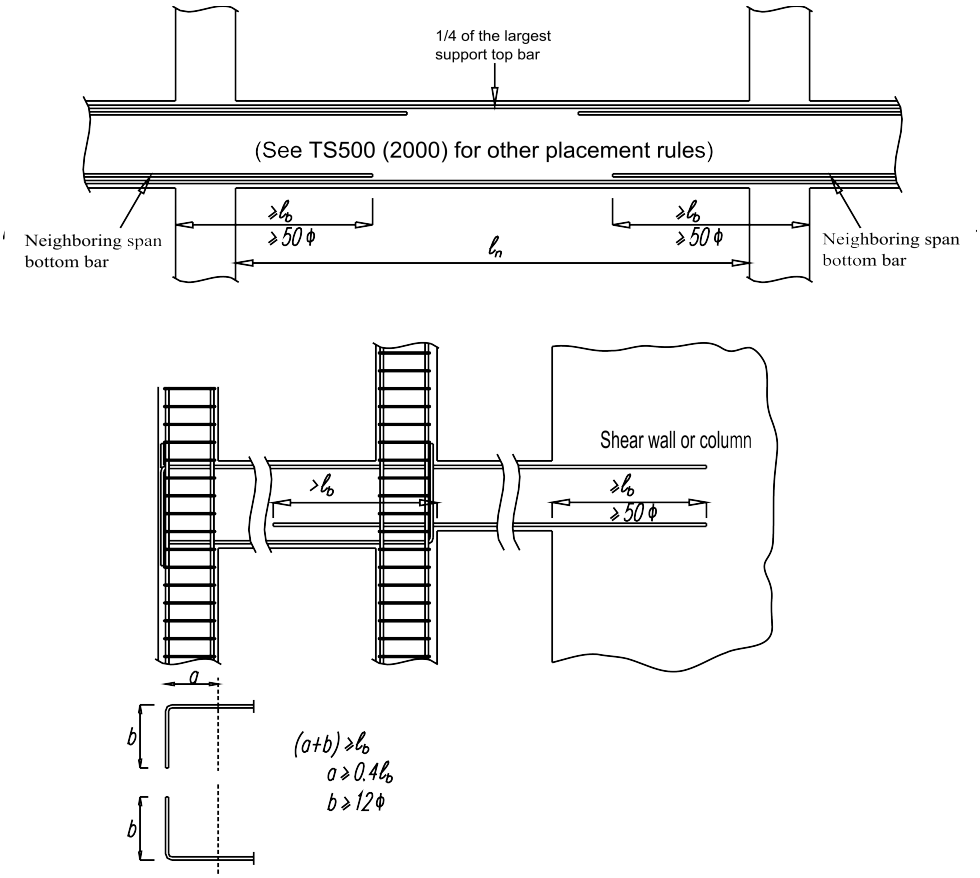


Figure 2.5: Development rules for longitudinal bars (DBYYHY, 2007)

reinforcement of the column. Rules for the development length of the bent longitudinal bars are presented in Figure 2.5. Total of the length of the horizontal part of the bar at the column section (a) and the length of the vertical part (b) must be greater than the value calculated with Equation 2.45. Horizontal length of the bent bar in the column must be greater than $0.4l_b$. Vertical length of the bar must be greater than 12ϕ. For shear walls and columns whose a dimension is greater than l_b and 50ϕ, bars can be developed straight into the supporting members.

If beams are continuous, bottom bars in the support section must be extended at least 50ϕ or l_b (Equation 2.45) from the column face into the next beam. If it is not possible to develop the bars into the next beam because of different section sizes or misaligned beam axes, then bars must be detailed according to the previous paragraph.

At critical locations like confinement regions or bottom bars of mid-span regions where lon-

gitudinal bars may yield, lap splicing is not allowed. In other regions, along the lap splices *special earthquake stirrups* must be placed. Spacing of the *special earthquake stirrups* must be less than $d/4$ or 100 mm.

2.5.2 TRANSVERSE BARS

In order to provide shear capacity, transverse reinforcement in forms of individual bars (stirrups, ties, etc.) or mesh bars can be used.

Spacing between transverse reinforcement (s) cannot be greater than half of the effective depth of beam ($s \leq d/2$). Additionally, if $V_d > 3V_{cr}$, spacing between transverse bars cannot be greater than $d/4$. At the end zones of frame beams, confinement regions covering a length of $2d$ from the support face must be provided. At the confinement regions, transverse bar spacing (s) must follow the rules below. ϕ_l is the diameter of the smallest longitudinal bar in the section.

$$\begin{aligned} s &\leq d/4 \\ s &\leq 8\phi_l \\ s &\leq 150\text{mm} \end{aligned} \tag{2.48}$$

In earthquake resistant design, the region between the support face and the point measured $2d$ from the support face is called the *Confinement Region*. Along the confinement region, *special earthquake stirrups* must be placed. The distance of the first stirrup from the support face must be less than 50 mm. Spacing of transverse reinforcement cannot be greater than the maximum spacing criteria described in the preceding paragraph. For normal ductility beams, in the confinement regions, spacing of transverse reinforcement cannot be greater than $1/3d$, $10\phi_l$, or 200 mm. In other regions of normal ductility beams, limitations defined in TS500 (2000) must be used.

Details for *special earthquake hoops* and *special earthquake crossties* is presented in Figure 2.6. Rules for special earthquake hoops and special earthquake crossties are given in the Turkish Earthquake Code (DBYYHY, 2007) as explained in the following paragraph.

Both ends of special earthquake hoops shall always be hooked 135^o. 90^o hooks can be used at one end of crossties. Lengths of hooks measured from tangent points shall not be less than 6ϕ or 80 mm for deformed bars. 135^o degree hooks shall be bent around a circle with at least 5ϕ diameter where ϕ is the diameter of the bar to be bent.

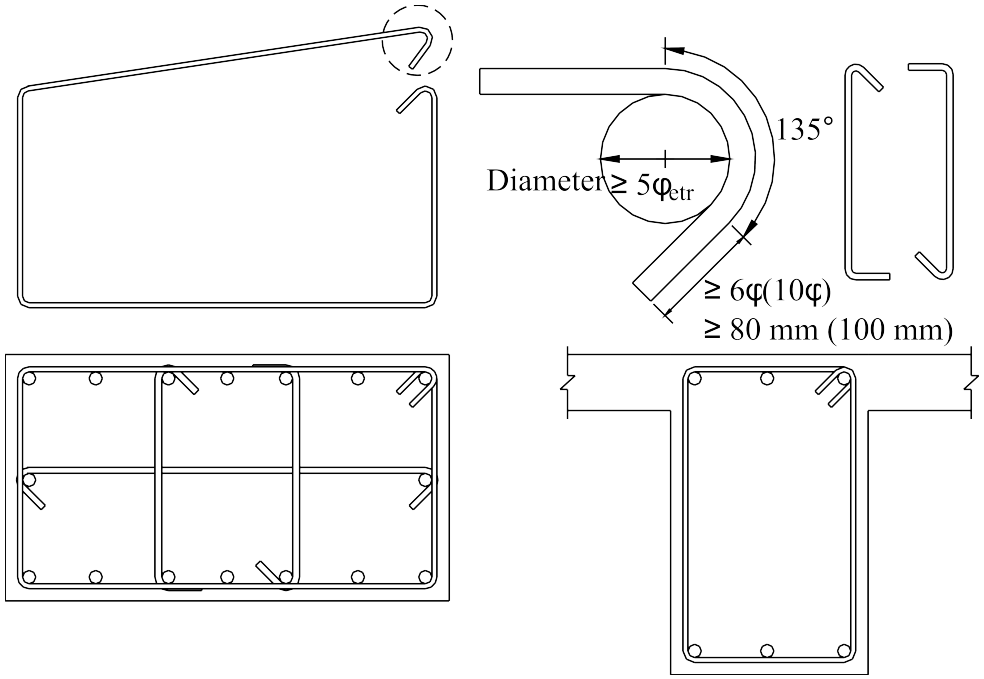


Figure 2.6: Details of special earthquake stirrups (DBYYHY, 2007)

2.6 FINAL CHECKS

After reinforcement bars are placed along beams to provide adequate capacity against design forces, final checks should be conducted for serviceability limitations. Besides providing capacity against collapse, reinforced concrete structural elements must be designed so that under design loads, excessive cracking, excessive deformations and excessive displacements must be prevented (TS500, 2000). Serviceability provisions of the Turkish Reinforced Concrete Code (TS500, 2000) are presented in the preceding sections.

2.6.1 DEFLECTION CHECK

Deflection check for beams that does not support or connected to deformation sensitive elements may be ignored if height to length ratio of the beam is greater than the values presented in Table 2.1 (TS500, 2000).

Table 2.1: Heigth/Length ratios of beams that does not require deflection checks (TS500, 2000).

	Simple Support	Exterior Span	Interior Span	Cantilever
Beam	1/10	1/12	1/15	1/5

When calculating immediate deflection of a beam under design loads, if maximum moment M_{max} does not exceed the cracking moment M_{cr} then, uncracked moment of inertia of the beam section can be used in the calculation. M_{cr} is calculated with Equation 2.50. Whereas, effective moment of inertia of the cracked section must be used if the beam has cracked ($M_{max} \leq M_{cr}$) (TS500, 2000). Effective moment of inertia of the cracked section can be calculated with Equation 2.49. Modulus of elasticity of concrete (E_c) is presented in Section 2.1.2.

$$I_{ef} = \left(\frac{M_{cr}}{M_{max}}\right)^2 I_c + \left[1 - \left(\frac{M_{cr}}{M_{max}}\right)^3\right] I_{cr} \tag{2.49}$$

$$M_{cr} = 2.5 f_{ctd} \frac{I_c}{y} \tag{2.50}$$

Cracking moment of the section is to be calculated by Equation 2.50. Average of the moment of inertias at the supports and spans must be used for calculation of the deflections for continuous beams. Moment of inertia at the support section must be used for cantilever beams.

It is stated in the Reinforced Concrete Code (TS500, 2000) that a method based on the principles of structural mechanics, which also takes reinforced concrete behavior into account can

Table 2.2: Limits of deflection (TS500, 2000).

Member and location	Cause of deflection	Span Length/Deflection
Roof member with no partition walls	Immediate deflection due to live loads	$l_n/180$
Floor member with no partition walls	Immediate deflection due to live loads	$l_n/360$
Roof or floor member with partition walls*	Sum of deflection due to permanent loads and deflection	$l_n/480$
Roof or normal story with partition walls	due to the remainder of live loads.	$l_n/240$

* : Members supporting or attached to partition walls or members likely to be damaged by large deflections.

be used in the calculation of immediate deflections. The nodal displacements are interpolated utilizing *Hermite interpolation functions* (Reddy, 1993).

2.7 INTERPOLATION OF FORCES AND DEFLECTIONS

2.7.1 INTERPOLATION OF FORCES

The internal forces are stored at various predefined locations along the beams. During the design it may be required to calculate forces at locations other than these predefined locations. In order to interpolate the forces at these intermediate points, 2^{nd} order polynomial interpolation functions were used.

2.7.2 INTERPOLATION OF DEFLECTIONS

Hermite interpolation function for a flexural 1D element is calculated with Equation 2.51 (Reddy, 1993). ϕ_i^e in the equation are known as *Hermite cubic interpolation (or cubic spline) functions*. Generalized displacements are denoted by u_i^e. Shape functions in local coordinates are presented in Equation 2.53.

$$w(x) = u_1^e\phi_1^e + u_2^e\phi_2^e + u_3^e\phi_3^e + u_4^e\phi_4^e = \sum_{j=1}^{4} u_j^e\phi_j^e \tag{2.51}$$

$$
\begin{aligned}
u_1^e &= w(x_e) && \text{: displacement at the } 1^{st} \text{ node of the beam.} \\
u_2^e &= \left(-\frac{dw}{dx}\right)\Bigg|_{x=x_e} && \text{: rotation at the } 1^{st} \text{ node of the beam.} \\
u_3^e &= w(x_{e+1}) && \text{: displacement at the } 2^{nd} \text{ node of the beam.} \\
u_4^e &= \left(-\frac{dw}{dx}\right)\Bigg|_{x=x_{e+1}} && \text{: rotation at the } 2^{nd} \text{ node of the beam.}
\end{aligned}
\tag{2.52}
$$

$$
\begin{aligned}
\phi_1^e &= 1 - 3\left(\frac{x}{L}\right)^2 + 2\left(\frac{x}{L}\right)^3 \\
\phi_2^e &= -x\left(1 - \frac{x}{L}\right)^2 \\
\phi_3^e &= 3\left(\frac{x}{L}\right)^2 - 2\left(\frac{x}{L}\right)^3 \\
\phi_4^e &= -x\left[\left(\frac{x}{L}\right)^2 - \frac{x}{L}\right]
\end{aligned}
\tag{2.53}
$$

Substituting the displacement and rotation values obtained from structural analysis as generalized displacements u_i^e into Equation 2.51 gives the equation of deflections on a beam. Equating the first derivative of Equation 2.51 $w'(x)$ to zero and solving for x yields the location of the maximum deflection on a beam, hence the maximum value of the deflection under design loads can be calculated.

CHAPTER 3

DATA MODEL

In this study, an object-oriented data model was developed for the design and detailing of reinforced concrete beams. The model not only holds the design specific information and detailed geometry of reinforcing bars for beams, but also provides object relationships that facilitates computations related to design process. The data model developed in this chapter is utilized both during the software implementation.

This chapter focuses on the motivation and the philosophy utilized during the design of the object libraries.

3.1 OBJECT ORIENTED PROGRAMMING - GENERAL

In object oriented programming, data and activities that are being modeled are grouped into virtual objects, where every object has properties and behavior just like real world objects (Liang, 2005). Using an object oriented approach improves software reusability and makes the program easier to develop and maintain. Every object has *fields*, or *attributes* (also called properties) whose values define its object state, and *methods* (also called functions) which define objects' behavior. For instance, a virtual object to model concrete material would have *fields* for storing its strength, modulus of elasticity, and unit weight. The stress-strain behavior of concrete would be described with a *method* which returns stress values for given strain values.

Fields and methods of general objects can be inherited into new objects to define more specific functionality. For example, a general rebar object can be created to define common properties of reinforcement types such as diameter, surface texture and length. Then, by inheriting

these common features and by defining more specific fields, objects for transverse bars and longitudinal bars can be programmed. This feature is known as *inheritance*.

Objects that inherit the same methods can be used interchangeably. Using the same method call definition, such objects can perform different behavior. This feature is known as *polymorphism*. For example, the algorithms of the methods that calculate the length of the rebars may differ for the longitudinal bars and the transverse bars. Since, the transverse bar and the longitudinal bar objects has a definition of length inherited from the general rebar class, a single method definition can be used to calculate the length of rebars.

Another useful feature of object oriented programming is *abstraction*. Abstraction is simplifying complex problems by modeling classes specific to the problem. Then, specific aspects of the problem can be attributed at the appropriate level of the abstraction. For example, transverse bar objects can be treated as rebar objects most of the time during design. Special attributes of transverse bars such as confinement parameters for example, can be described in the transverse bar class.

3.2 DATA MODEL ARCHITECTURE

Design of reinforced concrete frames usually initiates with the determination of framing system, initial member dimensions and material properties. If the design code requires certain limitations for member dimensions and material properties, then these requirements must be fulfilled. After the initial phase, the structure is modeled for structural analysis. The internal forces and displacements obtained from the structural analysis are utilized during the design and detailing process. Through design code regulations and application of principles of mechanics, required reinforcement amount to withstand design forces are calculated. Based on the calculated amounts, reinforcement bar diameters are determined and bars are placed through the concrete members. As a final phase, the designed structure is checked for conformance to serviceability criteria of the design codes. At any phase of the design process, it may be realized that the requirements of the design codes can not be satisfied with the initial assumptions. Then initial dimensions shall be reviewed, which requires re-analysis of the structural system. Hence, design and detailing of reinforced concrete frames is not a sequential process but an iterative process of analysis, design and code checking. The data model

that is capable of supporting design and detailing of reinforced concrete frames must also support such an iterative process.

In this study, in order to support such iterative process of design of reinforced concrete beams, a layered data model structure is developed. Figure 3.1 illustrates the layered architecture of the data model. As presented in the figure, there are two layers: The core layer called the *analysis layer* and the surrounding layer called the *design layer*. By definition, an object at an upper layer is allowed to use objects in the lower layer, whereas objects on the lower layer can not use objects in the upper layer. In other words, classes in the design layer uses the classes in the analysis layer to define their functionality. On the other hand, classes in the analysis layer are not aware of the the design layer. The contents of the layers are abstracted to such a level that easy extension of the data model to enable design and detailing of other reinforced concrete members such as columns, slabs, foundations, and shear walls can be possible in the future. Through the layered architecture of the data model,the analysis and design stages are conceptually separated while their relation was maintained.

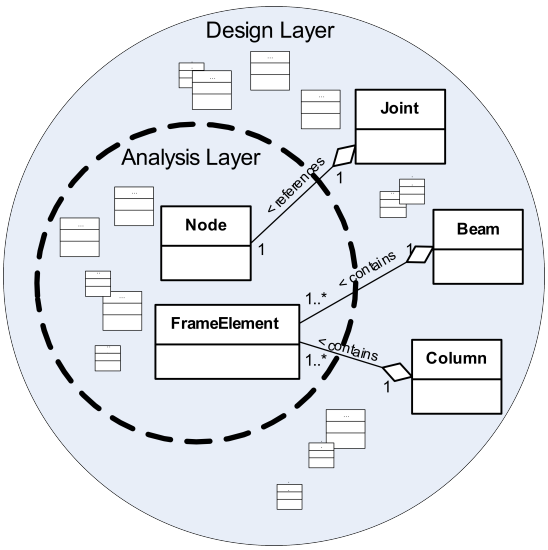


Figure 3.1: Data model architecture

3.3 ANALYSIS LAYER

The analysis layer contains a primitive geometry of the structure in the form of lines and nodes, and stores analysis results obtained from the solution of a 3D structural model. Cross sectional properties are also stored in this layer. The relation among objects are shown in Figure 3.2.

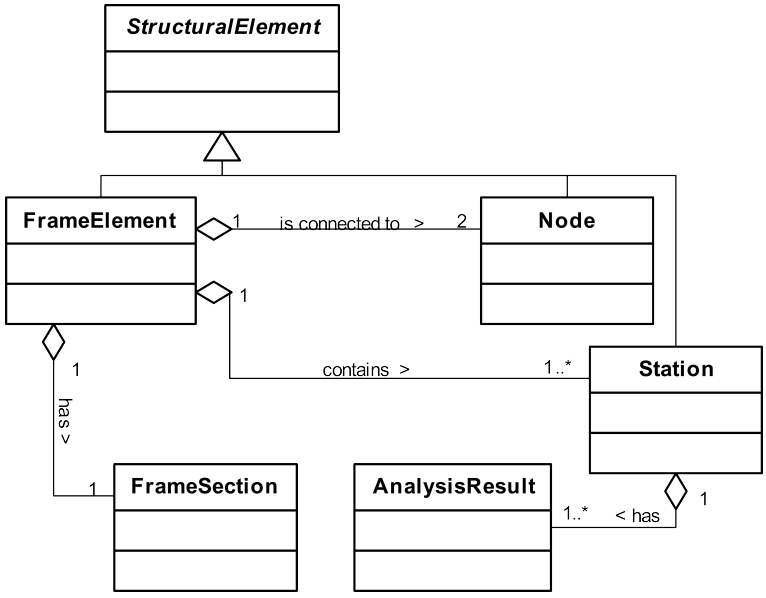


Figure 3.2: Class diagram of the analysis layer

In the proposed model, beams and columns were interpreted as 1D elements called *FrameElement* objects. *FrameElement* object has a field for storing a *FrameSection* to define the sectional properties. Every *FrameElement* object is connected two *Node* objects. *Node* object connects the *FrameElement* objects that have common end point coordinates. *Station* class was created to store the *AnalysisResult* and *Deflection* objects at various locations on the *FrameElement* objects. Generally the location of these station points are determined during the modeling process of the reinforced concrete structure. The structural analysis software which provides the model geometry and the analysis results to the developed framework yields the internal forces only at the stations and the deflections only at the nodes. The deflections are interpolated to the *Stations* by the methods of the *Beam* object which will be explained in the following section.

3.4 DESIGN LAYER

Around the *Analysis Layer* there is the *Design Layer*. The *Design Layer* has three main packages of classes. *Design Geometry Package* contains design objects such as *Beam*, *Joint*, and *Story*. Detailed geometry, properties and relations of reinforcing bars in reinforced concrete beams are defined in the *Reinforcement Package*. *Code Package* was developed to facilitate code based calculations and checks.

3.4.1 INTERPRETATION OF THE DESIGN GEOMETRY

The objects in the *Analysis Layer* may not be adequate for designing a structure and the realization of an actual structure's geometry during the design can be different than that of the analysis stage. For example, a beam can be modeled with several *FrameElements* during analysis, but the design practice requires that these sub-elements are designed as a single element. Moreover, in common practice of designing reinforced concrete frames, individual beams are usually designed as a part of continuous systems. Design of beam-column joints have special seismic design requirements, hence these connections must be specially attributed. In order to provide such improved definitions to the structural engineer, specific classes were defined to facilitate design. *Design Geometry Package* contains higher level objects, which groups, connects, or utilizes the objects in the *Analysis Layer* in order to improve the understanding of the structure programmatically and by the engineer. In this code package classes called *Beam*, *Column*, *BeamStrip*, *ColumnLine*, *Joint*, and *Story* are defined. Relations between these classes are shown in Figure 3.3.

The *Beam* class represents a single or a series of horizontally positioned *FrameElement* objects that are between two *Joint* objects. *Beam* object has methods for extracting the amount and pattern of reinforcement at any location on itself. It provides interpolation methods to calculate deflections and internal forces at any location. Every *Node* object that connects more than two beams or columns is defined as a *Joint* object. *BeamStrip* class was created to enable design of continuous beams as a single member and contains fields and methods for this purpose.

The *BeamStrip* stores an ordered array of *Joints* and *Beams* on itself. This way, design algo-

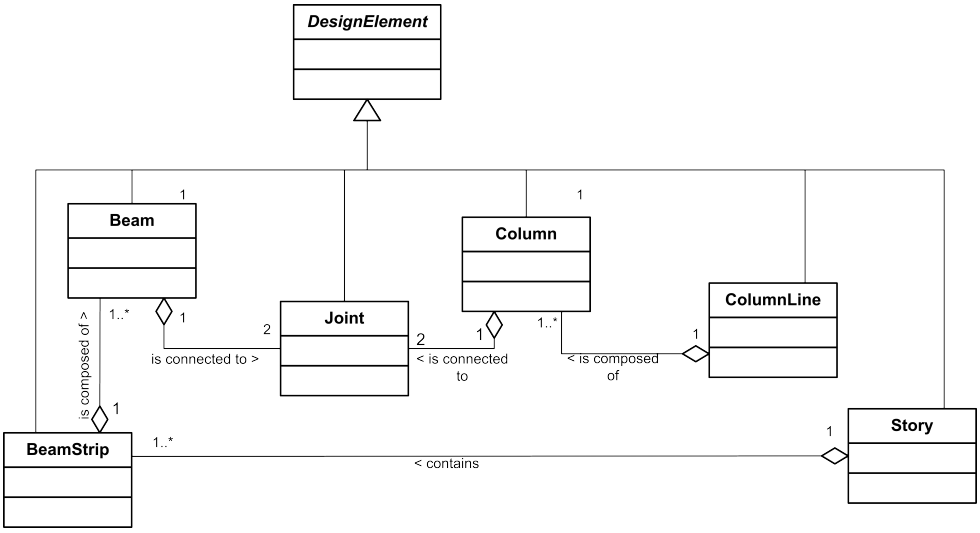


Figure 3.3: Design Geometry Package

rithms can move between *Beams* and *Joints* to perform calculations such as calculating the design forces at the two two faces of a column or extending longitudinal bars into the next beam.

Nodes, *FrameElements*, and *BeamStrips* that are on the same elevation are grouped into *Story* objects. *Story* object has a method for finding the continuous beams and creating *BeamStrip* objects on it.

Similarly, *Column* object represents *FrameElements* that are between two *Joints*. *ColumnLine* represents a series of *Columns* that are on the same grid line.

3.4.2 DESIGN COMPUTATIONS

As explained in Section 2.3.2, design codes require that the design forces are calculated at predefined sections on beams. Locations of these sections are dictated by the codes. In order to easily conduct design and detailing calculations at such predefined sections, *DesignSection* class was defined. Methods of the *DesignSection* can interpolate internal forces for any given load combination and calculate the maximum and minimum value of a force at that section. Also, they return the amount of reinforcement passing through that section. *DesignSection* also defines a region of a beam. These regions can be located at the left end, span, or right

end of a beam. This way, confinement regions at the beam ends can be defined. During seismic design, special detailing rules for confinement regions can be easily applied. Shear reinforcement calculated for a region can be distributed easily.

Longitudinal and transverse bar amounts and patterns are first calculated at the *DesignSections*. Then, the bars are extended and distributed through the *BeamStrips*. As explained, design and detailing of continuous beams is performed as a single unit through the *BeamStrip* objects. During design, it may be required that reinforcing bars to cross into other beams, be bent into column sections, or extend from one end of the beam to the other end of the beam. If the reinforcement amount or pattern at a section is changed during the design, the *DesignSection* that is responsible for that region or section must also be aware of the change. These detailing applications requires that every *Beam* shall know all *DesignSections* located on itself. Additionally, the *Beam* must know the *DesignSection* objects that are responsible for the specific regions of the *Beam*. In order to provide such features, relations presented in Figure 3.4 are created. *DesignSection* objects are stored on *Beam* objects. Also, *DesignSection* objects know on which *Beam* object they reside on. This way, detailing rules such as the minimum longitudinal reinforcement amount required at the supports as a percentage of the reinforcement amount at the spans can be provided.

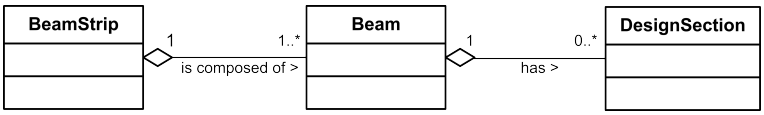


Figure 3.4: Relation of *DesignSection* class to *Beam* and *BeamStrip* classes

Table 3.1: Summary of fields and methods of the DesignSection class

Class Name: **DesignSection**	
Purpose: Provide a calculation point for the design and detailing of reinforced concrete beams.	
Field/Method Name	*Explanation*
LocationOnBeam	Location on the *Beam* object.
LocationOnStrip	Location on the *BeamStrip* object.
PositionStyle	Region of the *Beam* that the *DesignSection* object is in (left end confinement, span, or right end confinement).
DesignType	Used to define the type of the force (shear, flexure, etc.) that *DesignSection* is used for.
BarArea	Returns the area of reinforcing bars in that section.
DistI	Distance to the right end of the region that the *DesignSection* covers.
DistJ	Distance to the left end of the region that the *DesignSection* covers.
AnalysisResult	Used to store and retrieve the analysis results at that location of the beam.
ReinforcementPattern	Pattern of reinforcement stored at the *DesignSection*.

The summary of the fields and methods of the *DesignSection* class is given in Table 3.1. The beginning and end of the region covered by a *DesignSection* is defined by the *DistI* and *DistJ* fields of the object. For every type of force (shear, flexure, etc.) and the region on the beam, there is a dedicated *DesignSection* object. Hence, each *DesignSection* object has a *DesignType* field that is used to indicate the type of the force that *DesignSection* is used for the design of. *PositionStyle* field indicates the region that a *DesignSection* is responsible for. Possible values of the *PositionStyle* are *SupportI*, *SupportJ* and *Span* indicating left end confinement, right end confinement and mid-span regions during the shear design. There may be more than one *DesignSection* at a region. For instance, for the left end region of a beam, there may be two *DesignSection* objects: one for the flexural design, and one for the shear design. Independent of the *Stations* stored on the*Beams*, *DesignSection* objects also store the analysis results at that location of the beam. Values of the forces are interpolated at these sections and *DesignSection* provides methods to retrieve the values of the analysis results based on load combinations. The *DesignSection* object also has a reference to the reinforcing bars crossing that location on the beam.

3.4.3 CODE PACKAGE

Any structural design must follow specific rules defined by the design codes. Design codes may contain minimum requirements, material property and behavior definitions, design methods, and detailing rules. Additionally, different design codes may have common features or override the regulations of one another. For example, Turkish Earthquake Code (DBYYHY, 2007) defines the requirements to be applied in the seismic design where as Turkish Reinforced Concrete Code defines the general rules and the behavior of reinforced concrete. An abstraction to describe the common features of the design codes can help the easy development of design algorithms, and facilitate code based calculations.

List of the common features identified for design codes are presented in Table 3.2. A design code may define limits for the dimensions of structural elements or the amount of maximum force on an element. *CheckSize* and *CheckForce* methods are defined to check the dimensions and amount of force on an element. Material strengths may also have limitations. Reinforcement amount in a section, maximum deflection of a beam under service loads, and capacity of the beams also must be checked.

Table 3.2: Summary of the fields and methods in the abstract Code class.

Class Name: **Code**	
Purpose: Provide an abstraction for the common features of the design codes.	
Field/Method Name	*Explanation*
CheckSize	Checks the size of the beam.
CheckForce	Checks the force limits on the beam.
CheckRebarRatio	Checks the ratio of the reinforcing bars in a section.
CheckMaterialStrength	Checks the strength of the materials used in the design.
CheckCapacity	Check the capacity of a *DesignSection* on a beam.
CheckDeflection	Checks the maximum deflection on a beam.
CheckDevelopmentLength	Checks the development length of the reinforcing bars.
SetDesignSections	Calculates the location of the *DesignSection* objects on a beam and initializes the *DesignSection* objects.
SetDefaultCombinations	Sets the default combinations based on the design code.
GetDesignForce	Returns the design force at a *DesignSection*.
Capacity	Returns the capacity of a section.
BarRatio	Returns the limits of reinforcement amount in a section.

Design codes may have different approaches to design, thus they may utilize different load combinations or perform calculations at different sections. Turkish Earthquake Code performs shear design at the faces of the columns. On the other hand, Turkish Reinforced Concrete Code performs the shear design at a distance equal to the beam depth from the column face. *SetDesignSections* method calculates the location of the *DesignSections* on beams, whereas *SetDefaultCombinations* method combines load cases as specified by the design codes. *DesignForce* to be used in the design may be different that calculated in the structural analysis. For example, during seismic design, the shear force at the column faces are calculated by capacity design approach for high ductility beams. *GetDesignForce* method calculates the design force to be used in the design for a given *DesignSection*. For bending moment, this method returns the maximum value among the load combinations. On the other hand, for shear it returns the design force including the effect of concrete and bending capacities at the beam ends in seismic design.

While codes may define different rules, they may also override or use rules defined in other codes. Through object relations in the developed library it is possible to reuse algorithms and methods. For example, Turkish Earthquake Code and Turkish Reinforced Concrete Code have the same approach for the flexure design, but they have different approaches for shear design. During seismic design, for flexure calculations Turkish Earthquake Code can directly use the methods of the Turkish Reinforced Concrete Code and for shear design, it may define new algorithms. Similarly, both codes utilize rectangular stress block assumption for describing the stress-strain behavior of concrete under compression. It is adequate to write the algorithm to define this behavior in the Turkish Reinforced Concrete Code once and reuse it in the Turkish Earthquake Code. Similarly, calculation of section capacities and checking deflections are performed according to the Turkish Reinforced Concrete Code for both codes.

The common features of the design codes are abstracted into a general *Code* class. This class has no method implementation but only has method name definitions. By inheriting these abstract fields and methods in specific objects, specific design codes can be implemented. As explained in Section 3.1, objects that inherit the same object can be used interchangeably. Hence, through this abstraction, a structure designed with a specific design code can be checked for the compliance to another design code.

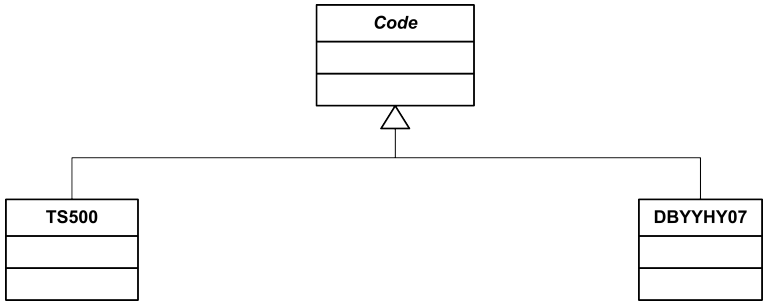


Figure 3.5: Abstract Code class and implemented concrete classes.

3.5 REINFORCEMENT CLASSES

In order to define detailed geometry and design specific information of reinforcing bars in reinforced concrete beams, several classes were developed in the *Reinforcement Package*. *Rebar* class was created to define a base class for reinforcing bars. Summary of the fields and methods of the *Rebar* class is given in Table 3.3. *Rebar* object has properties to store its diameter, surface texture, steel model, and cross section area.

Table 3.3: Summary of the fields and methods in the Rebar class.

Class Name: **Rebar**	
Purpose: Provide a general definition for reinforcement bar types in reinforced concrete beams.	
Field/Method Name	*Explanation*
Diameter	Diameter of reinforcement
SurfaceTexture	Surface texture of the reinforcing bar as defined in the design codes (Plain or deformed).
SteelModel	Material model to be used for calculating the stress-strain relationship or reinforcing bars (Not used currently in the design of beams but placed for future developments).
Area	Retuns the cross section area of the rebar.

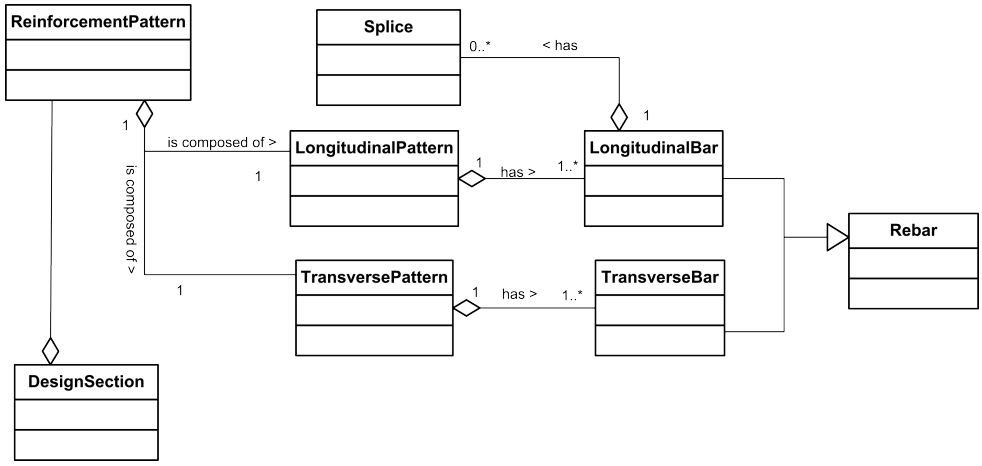


Figure 3.6: Reinforcement package

Object relations in the *Reinforcement Package* is presented in Figure 3.6. The general *Rebar* class was inherited into two child classes called *LongitudinalBar* and *TransverseBar*. These two classes define more specialized properties for the two kinds of reinforcement found in reinforced concrete beams: Flexural reinforcement that is placed longitudinally, and shear reinforcement that is placed transversely. Longitudinal and transverse bars in the beams are stored with the coordinates of the points on the bars. Each bar has a start point, an end point, and may have several bending points through its geometry. The origin of the coordinates of the *LongitudinalBar* objects is defined to be the first *Joint* on the *BeamStrip* that the bar is on. The origin of the coordinates of a *TransverseBar* is the upper left end of the beam section. These two special reinforcing bar classes has methods for extending, bending, and calculating the total length and weight of a bar. *Splice* class was created in order to be able to define lap splices. *Splice* class has fields for defining the location and the length of the splice. Additionally the *LongitudinalBar* class has methods to calculate lap splices on itself and stores the lap splices on it.

In order to be able to place the *LongitudinalBar* objects in a section easily, *LongitudinalPattern* object was defined. Similarly, *TransversePattern* object was defined to place the *TransverseBar* objects in a section and distribute the bars in a region of a beam that is defined by *DesignSection* objects. If, diameter of a *TransverseBar* is changed, locations of the *LongitudinalBars* in the section should be recalculated. Similarly, when the location of a *LongitudinalBar* in a section is changed, then the location of the *TransverseBar* that is surrounding it

should also be moved. The *ReinforcementPattern* object has methods to change the placement of *LongitudinalPattern* and *TransversePattern* objects in relation with each other. A change in the geometry, location, or size or a rebar in a section can be done via the *ReinforcementPattern* object easily.

3.6 DESIGN ALGORITHMS

The objects developed in this data model can perform several design calculations. These include calculating design forces, recommending reinforcement areas, calculating section capacities, distributing and placing rebars, and performing code checks. These tasks are performed by the responsible objects in the library, but in order to make recommendations related to the design, an intermediate object is needed. *Designer* class is developed for this purpose.

Designer class is an independent class and is not a part of the data model. It's purpose is to recommend reinforcement areas, select the diameter and number of rebars to be placed into a section, and perform detailing calculations such as extending and bending bars.

Summary of the fields and methods of the *Designer* class is presented in Table 3.4. The main field of the *Designer* is a code object, whose methods are called during a design process. *DesignBeamStrip* performs the complete design and detailing of a *BeamStrip* automatically.

DesignBeamStrip method performs the design and detailing of a *BeamStrip*. Pseudo code of the *DesignBeamStrip* method is presented in Figure 3.7. The algorithm traverses the *Joints* calculating and placing the longitudinal bars at the supports. Then, the *BeamStrip* is traversed once more to calculate the shear reinforcement and longitudinal reinforcement at the spans. The reason for traversing the *BeamStrip* two times is, in order to calculate the shear design force for seismic design, longitudinal bars must be known at the supports. First, the supports are designed for bending, then rest of the *BeamStrip* is designed.

Other methods of the *Designer* object are called during calculations. *RecommendBarArea* method returns a rebar area recommendation according to the design code. *MakeLongitudinalPatt* selects longitudinal bar diameters and places them in a section. *ExtendBendBars* method calculates the development length and bend points of the bars for placement along the *BeamStrip*. *MakeTransversePatt* select transverse bar diameter and spacing, and bends the

bars to fit into the section. *DistributeBars* calculates the number and location of transverse bars to be placed in a region of a beam defined by a *DesignSection*.

Table 3.4: Summary of the fields and methods in the Designer class.

Class Name: **Designer**	
Purpose: Calculate reinforcement areas and perform detailing tasks.	
Field/Method Name	*Explanation*
Code	Design code to be used in the design.
DesignBeamStrip	Performs the complete design and detailing of a *BeamStrip* automatically.
RecommendBarArea	Returns a recommendation for rebar area in a section.
MakeLongitudinalPatt	Selects the rebar diameters, and return a *LongitudinalPattern*.
MakeTransversePatt	Selects the rebar diameters, and return a *TransversePattern*.
ExtendBendBars	Calculates the development length and performs detailing tasks on longitudinal bars.
DistributeBars	Calculates the number and locations of transverse bars to be placed in a region of a beam

```
DesignBeamStrip( BeamStrip )
   for each Joint in BeamStrip
      RecommendBarArea for bending around the support
      MakeLongitudinalPatt
      ExtendBendBars
   end

   for each Beam in BeamStrip
      RecommendBarArea for shear at the left confinement
region
      MakeTransversePatt
      DistributeBars

      RecommendBarArea for bending at the span
      MakeLongitudinalPatt
      ExtendBendBars

      RecommendBarArea for shear at the span
      MakeTransversePatt
      DistributeBars

      RecommendBarArea for shear at the right confine-
ment region
      MakeTransversePatt
      DistributeBars
   end
end
```

Figure 3.7: Pseudo code of *DesignBeamStrip* method.

```
RecommendBarArea( DesignSection )
   if DesignSection has type V2
      GetDesignForce from the code object
      calculate and return reinforcement area
   end

   if DesignSection has type M3
      GetDesignForce from the code object
      get RebarRatio limits from the code object

      Check if only tensile reinforcement is adequate by the
method explained in Chapter 2

      if only tensile reinforcement is adequate
         calculate and return tension reinforcement area
      end

      else if compression bars are required
         Calculate tension and compression reinforcement
amount
         return reinforcement area
      end
   end
end
```

Figure 3.8: Pseudo code of *RecommendBarArea* method.

CHAPTER 4

WEB BASED MULTI USER ENVIRONMENT - XML WEB SERVICES

In the design and detailing of reinforced concrete frames, multiple engineers are involved. A web based environment can facilitate simultaneous design of a structure. In order to enable multiple engineers work on a single project simultaneously a client-server architecture was established. Additionally, rules and protocols were set up so that, design information was managed and latest updates were made available to project participants as needed. The framework utilizes XML (Extensible Markup Language) web services technology (Wolter, 2001) to build the web based environment. At the server side, all the design and detailing related information was kept as XML files and managed by the developed service. A brief review of XML technologies, system implementation, and how the information is handled and managed in the proposed framework are explained in this chapter.

4.1 XML TECHNOLOGIES

Among other technologies utilized in the implementation of web-based systems, for the last several years, XML related technologies have found wide support in the programming world (Wolter, 2001). Relatively easy implementation, simple concepts, and platform independence have served the reputation of XML related technologies (Bakis et al., 2007).

4.1.1 XML (EXTENSIBLE MARKUP LANGUAGE)

XML is a specification for creating custom markup documents. Every document that conforms the syntax and rules of XML Specification (Bray et al., 2008) is considered a well-formed XML document. Figure 4.1 presents a sample XML document containing a set of 3D *Point* objects. The first line of the sample XML document is the optional processing instructions, where the version of the XML Specification and encoding of the document are defined. XML documents have an object oriented structure is composed of blocks of data. Every data block in an XML document is enclosed between tags and is called a node. A tag can be any string that can be processed by the computer. Start tag of a node is enclosed between "<" and ">" characters, where end tag is enclosed between "</" and ">" characters. In the sample *PointDataSet* is the mandatory root node of the document. Nodes can contain other child nodes. Every *Point* node is a child node of the *PointDataSet* node. Nodes contain values and attributes. In the sample document presented in Figure 4.1, the *PointDataSet* stores 3 *Point* objects, where the *Point* object has the properties *XCoord*, *YCoord*, and *ZCoord*, and an identifier attribute *ID*. If, for example, the *XCoord* element had further properties, then these would be written enclosed between the start and end tags of the *XCoord* element.

```
<?xml version="1.0" encoding="utf-8" ?>
<PointDataSet>
  <Point ID="7485">
   <XCoord>10367.97</XCoord>
   <YCoord>12195.08</YCoord>
   <ZCoord>24000</ZCoord>
  </Point>
  <Point ID="7477">
   <XCoord>10369.36</XCoord>
   <YCoord>12195.11</YCoord>
   <ZCoord>24000</ZCoord>
  </Point>
  <Point ID="7448">
   <XCoord>869.3586</XCoord>
   <YCoord>11411.78</YCoord>
   <ZCoord>24000</ZCoord>
  </Point>
</PointDataSet>
```

Figure 4.1: Sample XML document.

XML documents can be queried by the XQuery and XPath languages (Stanek, 2002). An XML node with a specific value or a part of the XML document can be extracted by the standard querying constructs of XPath. Figure 4.2 presents an XPath query string which returns two *Point* nodes whose *ID* attributes have values of 7485 and 7448. The result of an XPath query can be processed further to extract values of the child nodes of the returned result node.

```
Point[@ID = '7485'] & Point[@ID = '7448']
```

Figure 4.2: Sample XPath query.

XML documents are processed by XML parsers (Stanek, 2002). There are two basic methods of parsing an XML document. The first one is the DOM (Document Object Model) parser which loads the entire document into memory and conducts the queries and other tasks in the memory. The major disadvantage of DOM is, as the document gets larger, the loading and processing time increases. Additionally, DOM parser requires greater memory allocation in order to parse a document. Other method of parsing an XML document is SAX (Simple API for XML) parser. SAX processes an XML document in an event-driven style. The document is traversed node by node as the caller program invokes read actions. In this model, the document is not loaded into the memory. Hence, SAX parser requires less memory usage than DOM parser. On the other hand, some tasks such as XPath queries needs to be able to access to any node at any time, which requires loading the document into the memory. DOM model supports such processes by design.

Using XML as a storage method or file transfer mechanism has several advantages over binary files and simple text files. First of all, XML has a predefined structure. There are certain rules defined in the XML Specification (Bray et al., 2008) that describes what is a well-formed XML. Secondly, XML files can be validated against an XSD (XML Schema Definition) schema which defines the structure of a specific XML document (Stanek, 2002). For example, it possible to define a specific XSD document which defines the structure of the sample presented in Figure 4.1. An XML parser can check the validity of the XML document against this schema and tell if it is structured correctly according to the schema. Hence, it is straightforward to evaluate an XML document as ill-formed or invalid if it is damaged while transferring or storage, which is not an easy task with binary and text files. Addition-

ally, XML is widely supported because XML is an open standard and is maintained by a community rather than a single software company.

A major disadvantage of XML is that, XML files can become very large easily because every node has a start and end tag. Size of the XML node of a value is always larger than the size of the stored value itself. For example, size of an XML document containing the internal forces of beams in an 8 story building consumes about 8 MB. On the other hand, if the internal forces were written in simple text files sequentially without using tags, it would not consume more than 4 MB. Still, storage and transfer issues of large documents can be resolved by utilizing stronger hardware. There are more advantages of utilizing XML as a storage and transfer mechanism than its disadvantages.

4.1.2 XML WEB SERVICES

An XML web service is "a software system designed to support interoperable machine-to-machine interaction over a network", as defined by W3C (World Wide Web Consortium) (Haas and Allen, 1994). An XML web service, or web service shortly, is a platform-independent, loosely coupled, self-contained, programmable, object oriented web-enabled application (Papazoglou, 2008). A web service can be described, published, discovered, coordinated, and configured using XML for the purpose of developing distributed interoperable applications. Web services can engage other services in order to complete a task, conduct transactions, or solve a complex problem. Web services expose their functionality over the internet using XML, and standard protocols such as HTTP (Hyper Text Transfer Protocol).

XML web services protocols, interfaces, and registry services enable applications to work cooperatively using the principles of loose coupling. An interface is the exposed methods and functionality of a web service, which are consumed by other applications. Having a neutral programmatical interface, which is not strongly tied to a particular implementation, is known as loose coupling. In order to achieve loose coupling, the web service interface is defined in a neutral manner that is independent of the underlying platform, the operating system, and the programming language the service is implemented in (Papazoglou, 2008).

An XML web service is a self-contained software module that performs a single task. The module describes its own interface characteristics, such as the operations available, the pa-

rameters, and access protocols in such a way that other software modules can determine what it does, how to use it, and what results to expect when it is called (Papazoglou, 2008). The functional and non-functional characteristics of a service are described by a standard description language called Web Services Description Language or WSDL.Functional characteristics define the behavior of the service, where non-functional characteristics define the properties of the hosting environment. A client application extracts the necessary information to be able to communicate with the web service by consuming the WSDL document (Wolter, 2001). WSDL is an XML document.

Applications communicate with a web service with SOAP (Simple Object Access Protocol) messages (Wolter, 2001). SOAP is a specification that defines the XML format for the messages that are exchanged between clients and the web service (Haas and Allen, 1994). Any XML document or fragment enclosed in SOAP elements is considered a valid SOAP message. SOAP messages have two main parts called the *Header* and the *Body* (Papazoglou, 2008). Contents of these parts are application specific and not dictated by the SOAP specification. *Header* contains the processing instructions of the SOAP message such as routing, delivery, authorization, and authentication information. *Body* part contains the actual message of the SOAP message. *Header* can be seen as the envelope of a traditional letter on which receiver and sender information are printed. Then, *Body* is the letter sent in the envelope.

SOAP messages are carried over the network using HTTP (Hyper Text Transfer Protocol) (Wolter, 2001). Other forms of transportation such as FTP, CD, or e-mails can also be used for exchanging SOAP messages but almost all XML web services use HTTP since it is the core protocol of the internet and most corporate networks have an infrastructure supporting HTTP (Wolter, 2001).

XML Web Services can be published over the internet by using UDDI (Universal Discovery Description and Integration). UDDI directories can be used to search and locate an XML web service for a specific purpose (Wolter, 2001). Publishing UDDI is optional.

4.2 WEB BASED SYSTEM ARCHITECTURE

XML Web Services technology was utilized in this framework to establish a client-server architecture in order to facilitate multi user simultaneous design and detailing of reinforced

concrete beams. The server contains the web service developed. Server is responsible for the storage of the information produced in the environment. All the computations are performed on the client side.

4.2.1 WEB SERVER

The server is responsible for the storage and management of information in the environment. All information is stored in XML format on the server side and no local copies of the files are allowed on the client side. Hence, there is always one real copy of each file. Table 4.1 and Table 4.2 present the lists of the XML files stored on the server. XML files are either created during the import or created during the design process. Imported files are those containing the model definition and the analysis results obtained from the structural analysis. Restraint conditions at the supports of the structure, local axis orientation of frames, and units of the analysis results are also stored. Reinforcement data, *BeamStrips*, *DesignSections*, and design parameters such as design code, diameters of bars to be used in the design are created during design. Contents of the XML files are structured so that they consume minimal space. As an example, length of the beams are not stored since it can be calculated from start and end coordinates of the nodes of the beams. Similarly, *Joint* objects are not stored in the database since they can be instantiated dynamically from the connectivity data.

In order to prevent duplicate efforts by engineers, every *BeamStrip* object is owned by a single engineer. The engineers are responsible for the design and detailing of the *BeamStrips* that they own. Ownership data of a *BeamStrip* is defined in the lock file of the *BeamStrip*. Lock files are named as *Story#*.Strip.*Strip#*.lock where # sign represents the *Story* and *BeamStrip* ids. These files contain the name of the owner of the *BeamStrip*, and when it is locked. As long as a *BeamStrip* is locked, it is considered as being designed, and it can not be modified by engineers other than its owner. Still, other engineers can view the reinforcement information of that *BeamStrip*. When an engineer finalizes his tasks with a *BeamStrip*, he releases the lock and tags the *BeamStrip* as finalized.

Table 4.1: Imported XML files.

File Name	*Explanation*
CurrentUnits	Stores the units used in the structural model.
Nodes	Coordinates of the nodes of the structural elements.
Frames	Connectivity data of the frames of the structural model.
Story	Story elevation and id data.
Restraints	Restraints at the nodes.
SectionProperties	Dimensions of beam sections.
SectionAssignments	Beam section assignment data.
FrameLocalAxes	Local axes orientation data.
BeamForces	Internal forces on the horizontal frame elements.
PointDisplacements	Displacements and rotations of nodes.
LoadCases	Load cases defined in the analysis stage and load combinations defined in the design stage.

Table 4.2: XML files created during the design.

File Name	*Explanation*
DesignParameters	Stores the design code specific parameters such as earthquake region and ductility level as explained in Chapter 2.
Diameters	Diameters of reinforcement bars that can be used in the design.
Story#.BeamStrips	*BeamStrip* objects at the story identified with *Story#*
Story#.Strip.*Strip#*	*DesignSection*, *ReinforcementPattern*, and connectivity data of *BeamStrip* identified with *Strip#* at story *Story#*.

4.2.2 CLIENT

On the client side, A GUI (Graphical User Interface) was developed to communicate with the server and perform design and detailing calculations. As stated previously, no information is kept on the client side. There are currently four main windows in the GUI as presented in Table 4.3. The main window is the start point of the client GUI. In the main window, project initialization tasks are performed, and the model can be visualized in 3D. Project preferences window prompts the user during the project initialization and design code, load combinations, and material strengths are selected in this window. StoriesAndStrips window draws the elevation view of the stories. *BeamStrips* on each story is listed. Engineers select the *BeamStrips* to work on in this window and proceed to the beam design window. In the beam design window design and detailing calculations are performed. *DesignSections* and analysis results can be investigated. Calls to the server for locking and unlocking of a *BeamStrip* is performed here.

Table 4.3: Windows available on the client side.

Window	*Explanation*
Main	Visualize structural model in 3D. Initialization of a new project.
ProjectPreferences	Preferences such as design code, load combinations and rebar diameters to be used in the design are defined in this window. Seen only during the import.
StoriesAndStrips	Visualize the plan view of the stories in the building.
BeamDesign	Beam design calculations are performed in this window. Visualize a *BeamStrip* object in elevation view. Visualize internal force diagrams and *DesignSections* See the details of the design conditions of a *DesignSection*, such as capacity and reinforcement amount. Lock/unlock a *BeamStrip* object for design.

```
SELECT GlobalX, GlobalY, GlobalZ
FROM [Objects and Elements - Joints]
```

Figure 4.3: An SQL query string to extract coordinate data of joints.

4.3 WEB BASED DESIGN AND DETAILING OF REINFORCED CONCRETE BEAMS

Web based design and detailing of beams of a reinforced concrete frame initiates with the import of the analysis results and structural model geometry obtained from the analysis stage. In the developed environment, there are no structural analysis tools. Hence, in this study, analysis results and model geometry are imported from ETABS software. After the analysis is completed, the model and the analysis results should be first exported into an intermediate format from ETABS. ETABS supports exporting from its internal data model to Microsoft Access Database. At this stage any machine readable format could be used as an intermediate format, but Access was preferred because Access databases can be queried with simple SQL (Structured Query Language) commands to extract required information. Figure 4.3 presents an SQL query string to extract point coordinates from the table named "Obkects and Elements - Joints". In the client GUI of the environment, information is imported by performing several other SQL queries on the Microsoft Access Database and the first set of the files listed in Table 4.1 are created. Created XML files are then sent to the web server for storage.

The general procedure of how a *BeamStrip* is designed in the framework is presented through an example design as illustrated in Figure 4.4. In this example, the design of a single *BeamStrip*, labeled as B101, is explained. During the design of a beam, information is exchanged between a client and the server when it is needed and in a per-request basis. This way, unnecessary utilization of the network connection is prevented. As illustrated in Figure 4.4, when the GUI is initialized for the design of a structure, only the geometry of the structure is downloaded. If an engineer (Engineer B) decides to work on a *BeamStrip* (B101), the analysis results, and if exits, design section and reinforcement information for the selected *BeamStrip* will be downloaded from the server. After that, the design and detailing tasks can be performed. In order to work on a *BeamStrip*, the client should obtain the ownership of that *BeamStrip*. In other words, as the ownership of B101 is requested by Engineer B, the

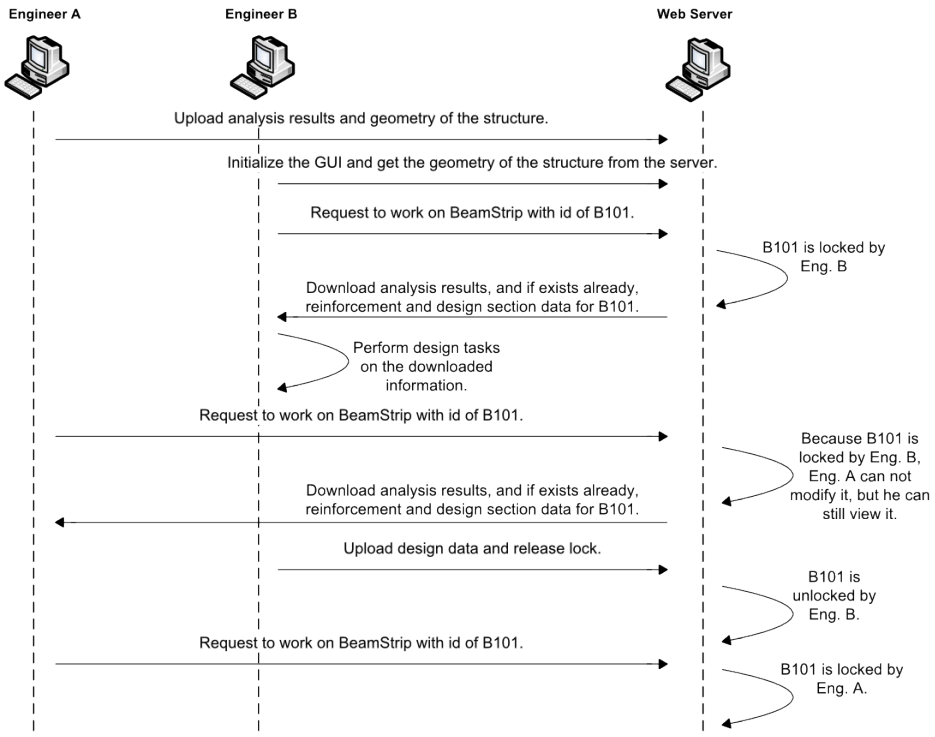


Figure 4.4: Simple sequence diagram for the web based design of a *BeamStrip*

beam strip B101 is locked for all other engineers' modifications, while they are still allowed to view the *BeamStrip*. When Engineer B finalizes his work with *BeamStrip* B101, the design related information such as the amount, size, and shape of rebars are uploaded to the server and the lock is released. Then, the strip becomes available for further modifications by other engineers. Between the lock/unlock requests, no active internet communication is required between the client and the server.

CHAPTER 5

DESIGN STAGES AND DEMONSTRATION

In this chapter, stages of designing a structure in the developed framework is explained through a sample demonstration of an actual model. The sample model is an 8 story building and is located in 2^{nd} earthquake region according to the Turkish Earthquake Code (DBYYHY, 2007). The building will be designed for normal ductility level. Structural system was modeled accordingly and solved in ETABS software.

5.1 PROJECT INITIALIZATION STAGE

Structural framing of a story of the building is presented in Figure 5.1. The 3D model of this building contains approximately 4000 nodes and 2400 frames. Microsoft Access Database of the model is 60 Mb.

The main window of the client GUI is presented in Figure 5.2. The project initialization stage starts with calling the put structure function call from the file menu on the main window. Microsoft Access Database to be imported is selected and the database is queried for required information. At the same time, extracted information is translated into XML in the developed data model format and object relations are created. The design geometry package calculates and initiates the *Beam*, *Joint*, and *BeamStrip* objects.

When the import is concluded, the user is prompted with a new window to select design parameters. The design code to be used in the design and code specific parameters are selected. *DesignSections* are calculated automatically by the design code at this stage. The design tab of the project preferences window and the selected design parameters are presented in Fig-

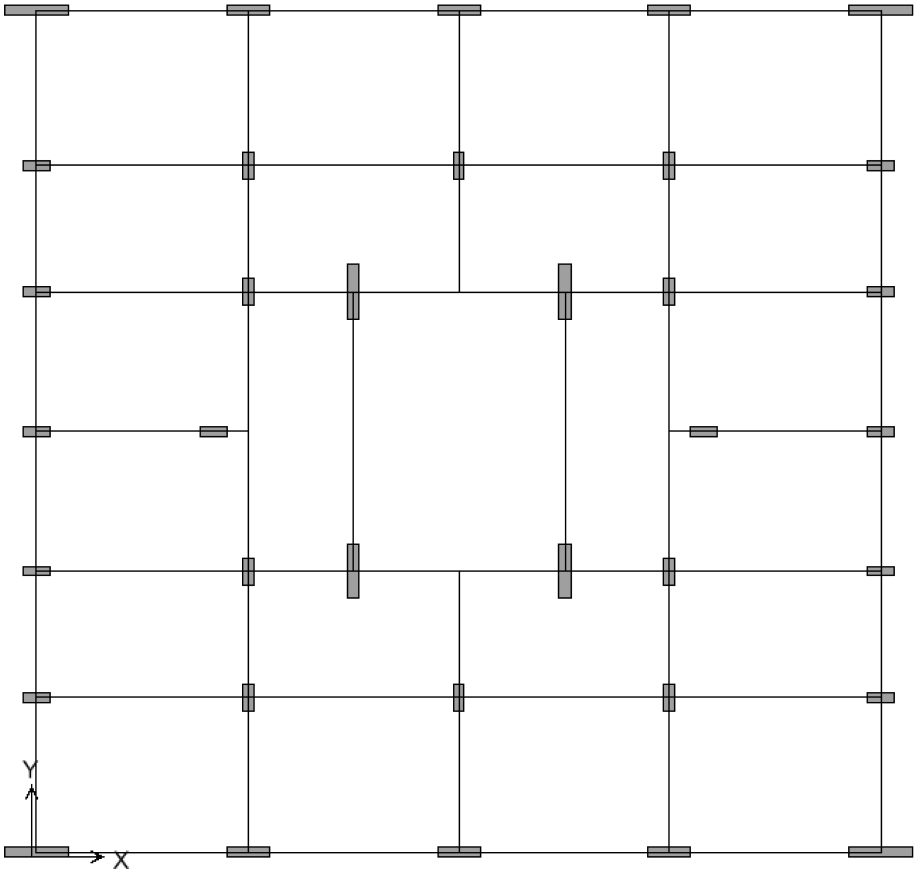


Figure 5.1: Structural system of the sample model

ure 5.3. The set defaults combinations button automatically calculates and initializes the load combinations that are defined in the selected design code as a minimum requirement. The user can define additional combinations. On the other tabs of the window, material strengths and the diameters of reinforcement bars that will be used in the design are selected. An initial check is performed on the model by the code for size, force, and material strength limitations. If any problem is found, the engineer is prompted.

Finally, XML files are sent to the server for storage. This stage is performed once. Translation and upload of the XML files to the server for the 8 story building model demonstrated here takes approximately 10 seconds on the same network. The XML files on the server for the initial stage consumes approximately 15 Mb. More than half of it is consumed by the internal forces.

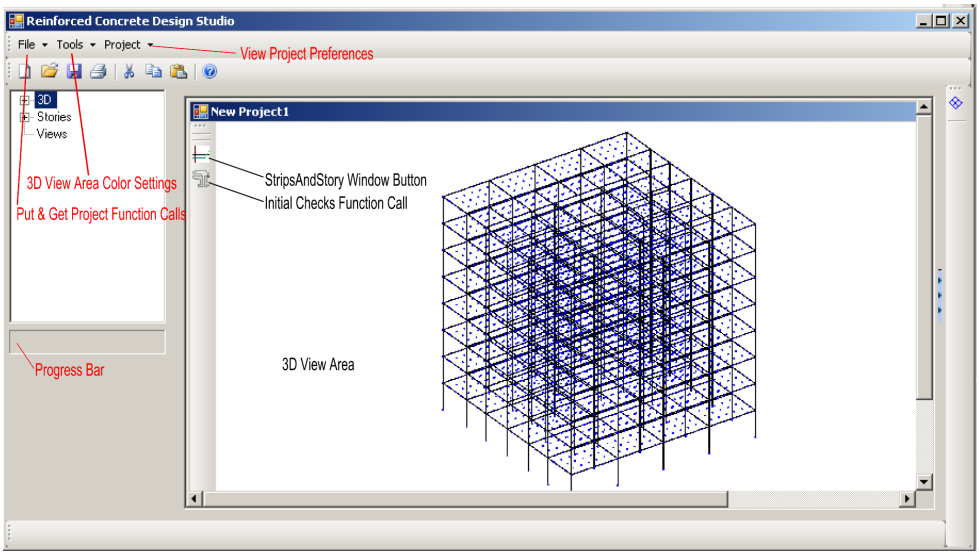


Figure 5.2: Main window of client GUI

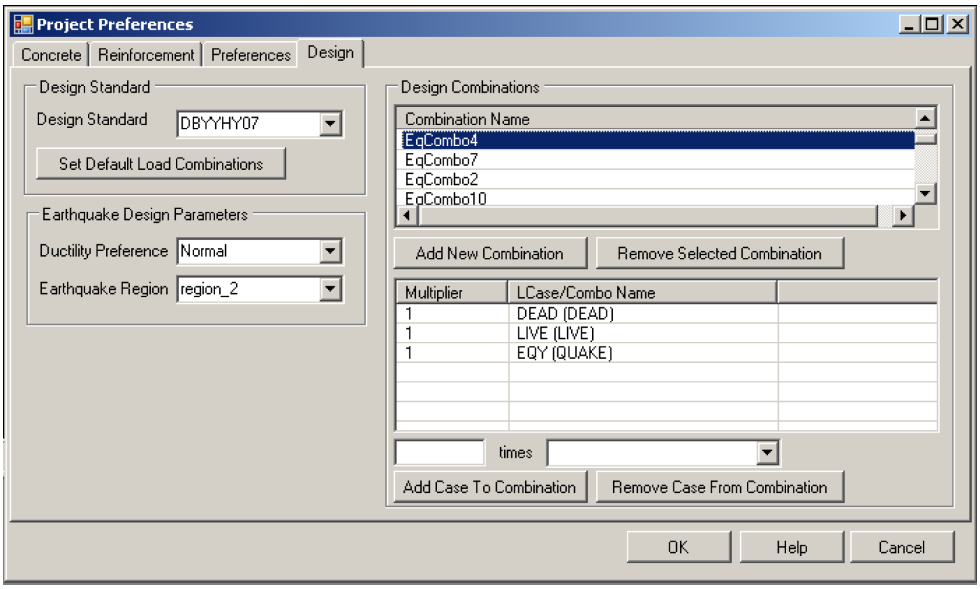


Figure 5.3: Project preferences windows

5.2 BEAM DESIGN

After the project initialization stage, an engineer can connect to the server and perform design calculations on the model. As the engineer calls the get structure function in the file menu on the main window, the model geometry is downloaded. Analysis results and reinforcement

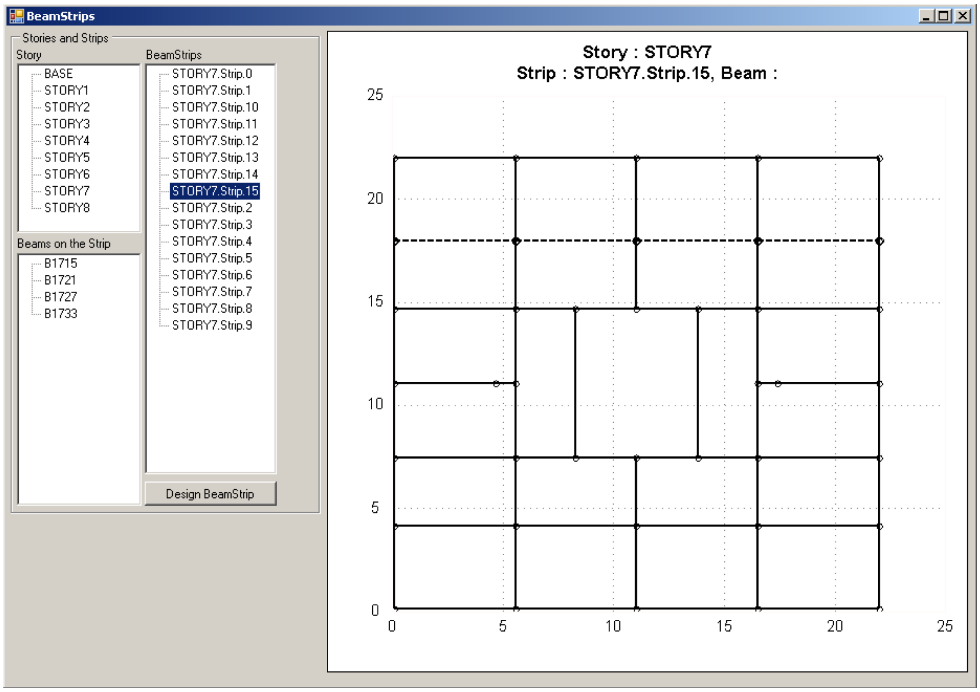


Figure 5.4: Plan view of the stories.

information is not downloaded yet in order to prevent unnecessary network usage. In the 3D view area of the main window, the structure can be visualized in 3D as presented in Figure 5.2.

When the strips and story button is clicked, StripsAndStory window appears. On this new window *BeamStrips* can be viewed on plan. StripsAndStory window is presented in Figure 5.4. The *Beams* on the selected *BeamStrip* are also listed. the selected *BeamStrip* is indicated with a dashed line. In this demonstration, the *BeamStrip* with the id of *Strip.15* on *Story* 7 will be designed.

The design beamstrip button proceeds to a new window on which the design and detailing calculations will be performed. As the design beam strip button is clicked on the StripsAndStory window, the client GUI connects to the server and downloads the analysis results, *DesignSection*, and reinforcement data, if present already for the *BeamStrip* selected. When the download is finished, which takes a few seconds for the selected 4 bay continuous beam, the BeamDesign window appears. In Figure 5.5, beam design window is presented.

At the top of the beam design window, there is a button for locking/unlocking the *BeamStrip*

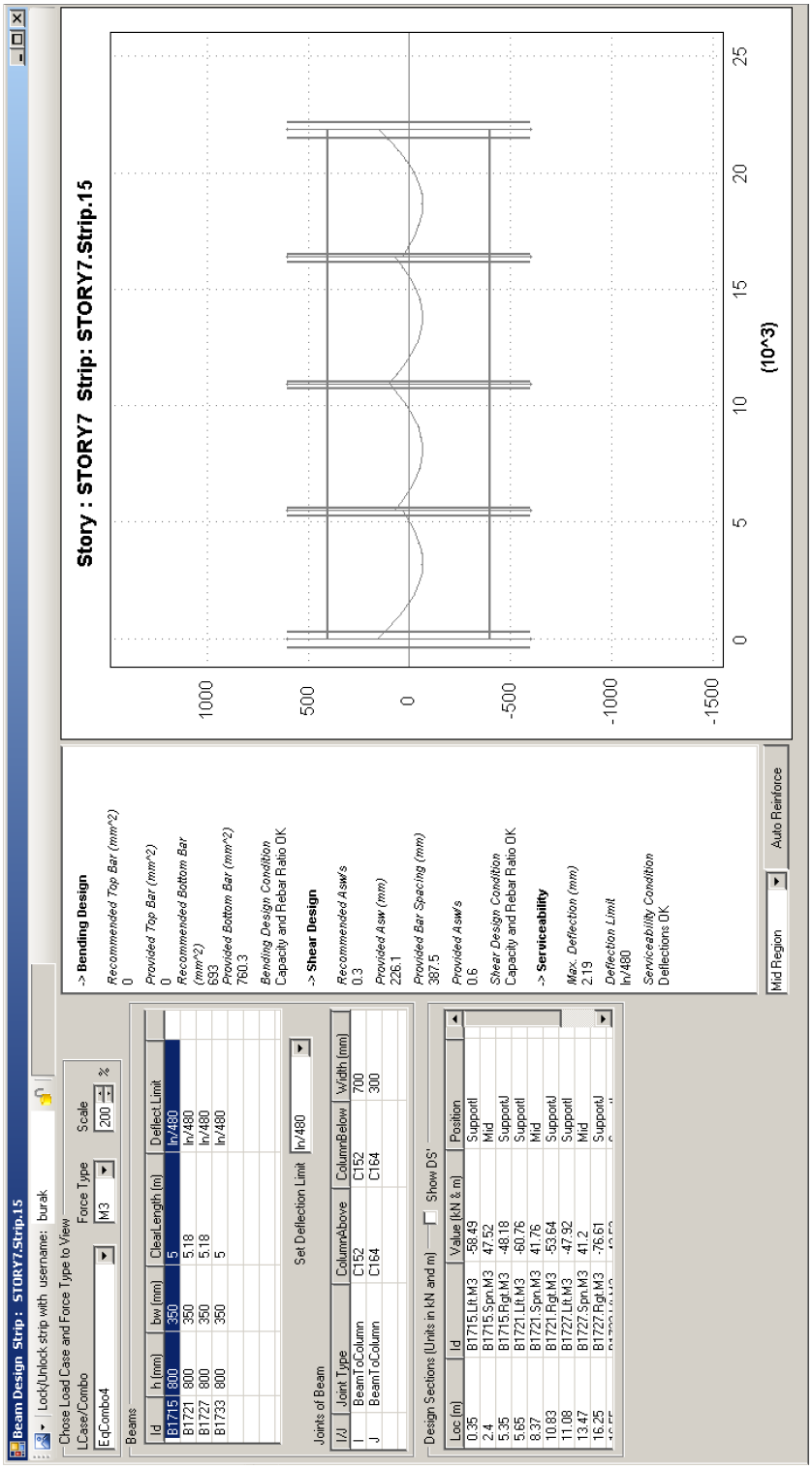


Figure 5.5: Beam design window

on the server. As the button is clicked, ownership request is processed for the user name entered. If the *BeamStrip* is already owned by another engineer, then the request owner is notified. Otherwise, *BeamStrip* is locked. Only its owner can modify a *BeamStrip*. Other engineers can only view.

On the left hand side of the window, beams and their geometrical properties on the strip are listed. Order of the beams are from left to right on the elevation view. By selecting a beam from the list, the engineer can see the id's of the column that are connected to the joints of the selected beam, joint connection types, and joint widths.In this case, the first beam with the id of *B1715* is selected. The joint on the left side of the beam has a width of 700 mm and it is connected to two columns.

Below the joints data, *DesignSections* on the *BeamStrip* are listed. By selecting different force types from the list located at the top left corner, the engineer can see the *DesignSections* for different force types. In Figure 5.5, M3 is selected which stands for bending moments. Their location relative to the center of the first joint of the *BeamStrip*, ids, design values, and their positions can also be seen. Design force values are calculated with the *GetDesignForce* method of the *Code* object.

In order to perform deflection checks, the engineer is given the option to select the limit of deflection for each beam. Deflection limits are selected from a list of values found below the beam list.

On the right hand side of the window, the elevation view of the *BeamStrip* is drawn. The *BeamStrip* can be drawn with different X and Y scales, hence for long *BeamStrips* such as this one, it is possible to view the complete *BeamStrip*. Additionally, force diagrams are drawn on the elevation view for the load combinations and force types.

In the middle of the window, design information is shown. In order to reinforce the *BeamStrip*, AutoReinforce button is clicked. This button calls the *DesignBeamStrip* method of the *Designer* object which calculates the reinforcement amounts and details the bars acoording to the code regulations. Finally, deflection limits are checked. Details of the calculations can be viewed for regions (left confinement, mid, right confinement) of the beams by selecting a beam from the beam list and a region from the box found below the design information table.

When the engineer finalizes his work he unlocks the *BeamStrip* by clicking the unlock button.

Calculated reinforcement information is uploaded to the server and stored.

5.3 DXF EXPORT

In the developed framework, it is possible produce CAD drawings of the *BeamStrips* and the contained reinfrocement in 3 dimensions. Figure 5.6 presents several views of the designed and detailed beam strip.

Figure 5.6: CAD output.

CHAPTER 6

CONCLUSION

Structural design of reinforced concrete frames is a complex process which requires multiple engineers to work on a project and large volume of information is produced and exchanged during the design tasks. At the initial stage the structure is modeled and analyzed for assumed member dimensions and framing. Analysis results obtained from the analysis stage are checked against design code limitations as explained in Chapter 2. Then the reinforcement design and detailing tasks is performed. Required reinforcement areas are calculated and bar diameters and placement is specified according to the rules as specified in the design codes. Finally, the structure is checked for serviceability limitations. At any stage of the design process, it may be realized that the requirements of the design codes can not be satisfied with the initial assumptions. Initial member sizes and framing shall be required to be revised which requires re-analysis of the structure with new assumptions.

Additionally, design of reinforced concrete structures requires involvement of multiple engineers to contribute to a single project. All the information between the engineers must be coordinated to prevent data losses, economical losses, and time delays associated with information exchange failures such as missing updates. Information exchange between participants in a structural design process can be facilitated effectively by utilizing computer technologies.

An information system that is capable of serving the management of information and engineering tasks during the design process of reinforced concrete beams should also support the special information needs that are described in the preceding paragraph. Thus, a special data model is needed to define the detailed geometry of the reinforcement and design specific information.

In this study, a data model that is capable of describing the detailed geometry of reinforcement in reinforced concrete beams developed. The data model is also capable of supporting the design calculations and facilitating an improved geometrical definition of the frame geometry to the engineer. Three main object oriented code packages have been developed. The first package is responsible for defining basic geometrical definitions for the design process such as joint, beam, story, and beam strip. Second package supports design calculations by providing special objects in order to facilitate design calculations at special design sections on the beams. The third package facilitates the design code based design of reinforced concrete beams. Common design and detailing rules that are accepted by the Turkish design codes (TS500, 2000; DBYYHY, 2007) are abstracted in to a general *Code* class and two implementations have been made for TS500 (TS500, 2000) and Earthquake Code (DBYYHY, 2007). The data model is programmed in such a way that it has a layered structure: Analysis Layer and Design Layer. In the analysis layer analysis results and a primitive geometry of the structure in the form of lines and nodes are contained. In the design layer, design specific information is contained. These two layers are separated conceptually while their relationship is maintained. Objects in the design layer uses the objects in the analysis layer to define their functionality. While, objects of the analysis layer are not aware of the design layer. The object libraries are extendable so that new objects can be defined in the future to facilitate the design of other reinforced concrete members such as shear walls, columns, and slabs.

Through utilization of XML Web Services technology (Wolter, 2001), a web based system has been implemented to facilitate simultaneous multi user design of reinforced concrete beams. A client and server relationship has been established to perform the design calculations and to store the information. On the client side, only calculations were performed. On the other hand, the information was kept in XML format on the server side. During the storage the developed data model has been employed as the main data model of the web based system. On the client side, several windows have been programmed. In these windows, the model can be visualized in 3D, on the plan view, or on the elevation view. Beam strips that are calculated automatically viewed. Design calculations are performed in a separate window where the design sections

The design of a structure initiated with the upload of the building geometry and the analysis results into the server. While uploading, the client side software calculates the beam strips, design sections, and prompts the user for the selection of design preferences. After the initial

upload, the engineers can connect to the server and perform calculations. In order to prevent duplicate efforts by the engineers, engineers lock the beam strips that they design on the server side. While a beam strip is locked, it can not be modified by other engineers, while they are still be able to view it. After an engineer finalizes the design of a beam strip, he uploads the new reinforcement information and releases the lock. In order to prevent excessive network usage, client only requests an information when needed. For instance, when the client GUI is first initialized, only building geometry is downloaded. Then, when an engineer requests to work on a beam strip analysis results, design sections, and reinforcement information of only that beam strip is downloaded.

The efficiency of the developed data model and the web based environment is investigated in an actual implementation. An algorithm that automatically designs and details a *BeamStrip* is developed. The algorithm is capable of recommending reinforcement bar areas for given *DesignSections*, selecting the rebar diameters and bar placing, and calculating extension and development length of rebars. The algorithm utilizes the object relations extensively during the calculations. It has been evaluated that, through such a data model it is possible to perform design and detailing tasks for reinforced concrete beams.

Design of a four bay *BeamStrip* of an eight story building is demonstrated in the web based environment. Despite the large size of the model, the network utilization and download times remained limited to a few seconds. The request based optimization of the network usage played an important role in limiting the transfer times. The web based system is utilized only for computers that are on the same network. The transfer times may take longer for servers and clients located on different networks but still with the advent of faster network technologies these borders can be broken. On the other hand, engineers generally work in the same office, hence the computers are located on the same network. The sample utilization demonstrated in this thesis reflects the general case.

6.1 FUTURE RECOMMENDATIONS

In the future the developed web based framework for the design and detailing of reinforced concrete beams can be enhanced in several ways. Firstly, the data model can be enriched by the addition of other reinforced concrete structural elements such as columns, slabs, shear

walls, and foundations. This way, a complete design and detailing environment can be provided to the design engineers. Secondly, an analysis engine can be integrated to the framework to perform structural analysis. Hence, iterative process of analysis, design and code checking can be completely facilitated within the framework. Finally, rules and protocols for the multi user simultaneous design can be developed further to include the roles of other participants such as contractors and managers into the design process.

Bibliography

Autodesk, 2009. Autocad buzzsaw. URL www.buzzsaw.com. Last accessed on January 2009.

Bakis, N, Aouad, G., Kagioglou, M., 2007. Towards distributed product data sharing environments - progress so far and future challenges. *Automation in construction*, 16(5):586–595.

Bentley, 2009. Projectwise. URL www.bentley.com. Last accessed on January 2009.

Bjork, B. C., 1989. Basic structure of a proposed building product model. *Computer Aided Design*, 21(2):71–78.

Bray, Tim, Paoli, Jean, Sperberg-McQueen, C. M., Maler, Eve, Yergeau, FranÃ§ois, 2008. *Extensible Markup Language (XML)*. URL http://www.w3.org/TR/REC-xml/. Last accessed on January 2009.

Chassiakos, A. P., Sakellaropoulos, S. P., 2008. A web-based system for managing construction information. *Advances in Engineering Software*, 39(11):865–876.

Chen, Hung Ming, Lin, Yu Chin, Chao, Yi Feng, 2006. Application of web services for structural engineering systems. *Journal of Computing in Civil Engineering*, 20(3):154–164.

Chen, Hung Ming, Tien, Hung Chun, 2007. Application of peer-to-peer network for real-time online collaborative computer-aided design. *Journal of Computing in Civil Engineering*, 21(2):112–121.

Clark, James, 1999. *XSL Transformations (XSLT)*. URL http://www.w3.org/TR/xslt. Last accessed on January 2009.

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

Eastman, Charles, Jeng, Tay Sheng, 1999. A database supporting evolutionary product model development for design. *Automation in construction*, 8:305–323.

Eastman, Charles M., 1999. *Building Product Models: Computer Environments Supporting Design and Construction*. CRC Press, USA. ISBN 0849302595.

Eastman, Chuck, Teicholz, Paul, Sacks, Rafael, Liston, Kathleen, 2008. *BIM Handbook - A guide to building information modeg for owners, managers, engineers, and contractors*. John Wiley and Sons, Inc.

Ersoy, Ugur, Ozcebe, Guney, 2001. *Betonarme*. Evrim Yayinevi, Ankara.

Faraj, I., Alshawi, M., Aouad, G., Child, T., Underwood, J., 2000. An industry foundation classes web-based collaborative construction computer environment: Wisper. *Automation in Construction*, 10:79–99.

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*. NIST (National Institute of Standards and Technology).

Haas, Hugo, Allen, Brown, 1994. *Web Services Glossary*. W3C (World Wide Web Consortium). URL `http://www.w3.org/TR/ws-gloss/. Last accessed on January 2009`.

Han, Charles S., Kunz, John C., Law, Kincho H., 1998. Client/server framework for on-line building code checking. *Journal of Computing in Civil Engineering*, 12(4):181–194.

Han, Charles S., Kunz, John C., Law, Kincho H., 1999. Building design services in a distributed architecture. *Journal of Computing Civil Engineering*, 13(1):0–0.

IAI, 2006. Ifc 2x edition 3 model definition. URL `http://www.iai-international.org/Model/R2x3_final/index.htm. Last accessed on January 2009`.

Liang, Daniel Y., 2005. *Introduction to Java Programming - Comprehensive Version*. Pearson Education International.

NIST, 1999. Iges. URL `http://www.nist.gov/iges. Last accessed on January 2009`.

Papazoglou, Michael P., 2008. *Web Services: Principles and Technology*. Prentice HallPearson Education Limited.

Pope, Alan, 1998. *The CORBA reference guide: understanding the common object request broker architecture*. Addison-Wesley.

Reddy, J. N., 1993. *An introduction to the finite element method*. McGraw-Hill International, Inc.

Rosenman, M. A., Smith, G., Maher, M. L., Ding, L., Marchant, D., 2007. Multidisciplinary collaborative design in virtual environments. *Automation in Construction*, 16(1):37–44.

Stanek, William R., 2002. *XML*. Microsoft Press.

TC184/SC4. The step (iso 10303) product data representation and exchange. URL `http://www.tc184-sc4.org/SC4_Open/SC4 Legacy Products (2001-08)/STEP_(10303)/. Last accessed on January 2009`.

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

Watson, A., Crowley, A., 1994. Cimsteel integration standards. In R. J. Sherer, editor, *Proceedings of ECPPM'94*. The First European Conference on Product and Process Modeling in the Building Industry, Dresden, Germany.

Wolter, Roger, 2001. Xml web services basics. URL `http://msdn.microsoft.com/en-us/library/ms996507.aspx. Last accessed on January 2009`.

Yasaka, Ayako, Kataoka, Hiromi, Takeda, Makoto, Usami, Masato, Yabuki, Nobuyoshi, Matumoto, Norihide, Furukawa, Satoru, Mogi, Yoshihiro, Adachi, Yoshinobu, 2006. The development of the reinforced concrete structural model on ifc specification. In *Proceedings of International Conference on Computing in Civil and Building Engineering*.

APPENDIX A

OBJECT ORIENTED DESIGN WITH UML

UML (Unified Modeling Language) is a visual method for modeling object oriented relations in a software (Liang, 2005). Most used type of diagram of UML is the class diagrams where the static relations between objects are modeled.

A.1 CLASS DIAGRAMS

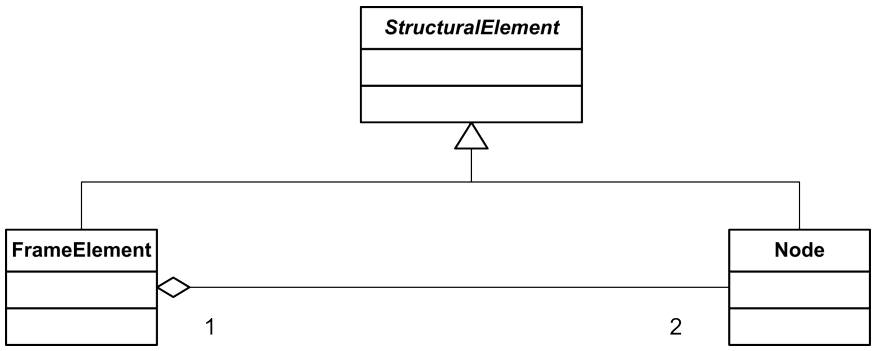


Figure A.1: A sample class diagram

In a UML class diagram, a class is denoted by a rectangle with the name of the class at the top of the rectangle as presented in Figure A.1. The name of an abstract class in which there are no method implementations but only method and field names are defined is written in italic fonts.

Inheritance is defined by an arrow connecting the parent class and the child class. The end of the arrow points the parent class. In the example, *StructuralElement* class is inherited into two concrete classes called *FrameElement* and *Node*.

A class can use another class to define its functionality. This relation is called composition. In the example, *FrameElement* class has a composition relation with the *Node* class. In composition relation, the classes are connected with a line with a diamond on the using classes side. The number on the two sides of the composition relation denote the multiplicity of the relation. In Figure A.1, a *FrameElement* object has two *Node* objects.

APPENDIX B

UML OF THE DATA MODEL

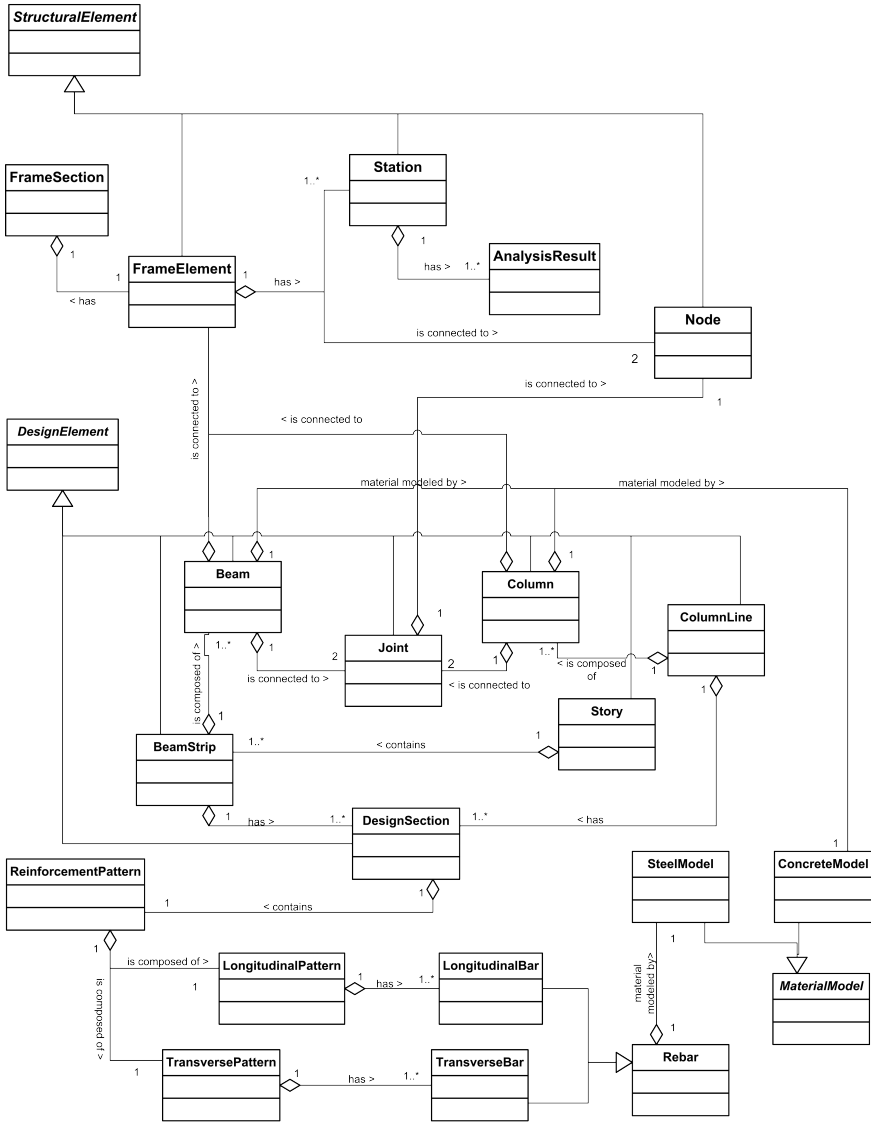


Figure B.1: UML of the data model