UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ESTRUTURAS

AUTOMAÇÃO DO PROCESSO DE DETALHAMENTO DE TORRES METÁLICAS VIA TECNOLOGIA CAD

Aluno: Henrique Dias de Oliveira Gontijo
Orientador: Prof. José Ricardo Queiroz Franco
Co-orientador: Felício Bruzzi Barros

Belo Horizonte, Agosto de 2010

UNIVERSIDADE FEDERAL DE MINAS GERAIS
ESCOLA DE ENGENHARIA
CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ESTRUTURAS

"AUTOMAÇÃO DO PROCESSO DE DETALHAMENTO DE TORRES METÁLICAS VIA TECNOLOGIA CAD"

Henrique Dias de Oliveira Gontijo

Dissertação apresentada ao Curso de Pós-Graduação em Engenharia de Estruturas da Escola de Engenharia da Universidade Federal de Minas Gerais, como parte dos requisitos necessários à obtenção do título de "Mestre em Engenharia de Estruturas".

Comissão Examinadora:

Prof. Dr. José Ricardo Queiroz Franco
DEES – UFMG – (Orientador)

Prof Dr. Felício Bruzzi Barros
DEES – UFMG

Prof. Dr. Armando Cesar Campos Lavall
DEES – UFMG

Prof. Dr. Philippe Remy Bernard Devloo
UNICAMP

Belo Horizonte, Agosto de 2010

DEDICATÓRIA

Ao meu pai, grande incentivador e ídolo.

A minha mãe pelo amor incondicional e conselhos.

A minha irmã pelo companheirismo em todas as horas.

A toda minha família que amo tanto.

Aos meus amigos por estarem sempre presentes.

AGRADECIMENTOS

Sincero agradecimento a todos que de alguma forma contribuíram para a elaboração deste trabalho e em particular:

- Ao Professor José Ricardo Queiroz Franco pela orientação, incentivo e os ensinamentos durante o desenvolvimento do trabalho.

- Aos professores Regina e Felício pelos ensinamentos e conselhos.

- Ao Leão, Tadeu, Emmo e Gilberto pelos ensinamentos.

- A CAPES – Coordenação de Aperfeiçoamento de Pessoal de Ensino Superior, pela concessão de Bolsa de Estudos durante o período de realização deste trabalho.

- A empresa CR Gontijo Engenharia de Projetos e todos seus integrantes por todo o suporte técnico oferecido.

RESUMO

O presente trabalho apresenta o projeto de desenvolvimento de um sistema computacional, denominado TowerCAD, para automação da etapa de detalhamento do processo de fabricação de estruturas de torres metálicas através da integração das tecnologias CAE (Computer Aided Engineering), CAD (Computer Aided Design) e CAM (Computer Aided Manufacturing). O sistema TowerCAD,que é constituído de um aplicativo CAD e de um banco de dados, detalha uma parte da estrutura denominada perna e foi elaborado considerando o futuro desenvolvimento do detalhamento das outras partes da estrutura.

Além da tecnologia CAD, o TowerCAD está sendo implementado com auxílio dos paradigmas da Programação Orientado a Objetos e das tecnologias de Banco de Dados. A tecnologia .NET, desenvolvida também para ser uma interface de programação (API) do AutoCAD, foi utilizada para desenvolver as interfaces para importação e exportação de dados, criação de comandos, gerenciamento, edição e manipulação dos dados no banco de dados. O banco de dados é acessado pelo aplicativo através das classes ADO.NET, a linguagem padrão para banco de dados SQL e a linguagem de programação C#. A plataforma de representação gráfica utilizada é o "*software*" AutoCAD.

Uma importante contribuição desse trabalho é a utilização de conceitos e fundamentos do padrão/arquitetura MVC (Modelo-Visualização-Controle) no desenvolvimento do sistema TowerCAD. A arquitetura de três camadas do MVC para a interface do usuário permite separar a informação, o controle e a representação gráfica.

O componente **Modelo,** que representa os dados e suas regras de acesso e a manipulação, está armazenado no banco de dados. O componente **Visualização**, nesse caso será representado pela plataforma gráfica AutoCAD, que apenas executa a apresentação de dados conforme solicitado pelo usuário. O **Controlador** interpreta as ações do usuário alterando ou não o estado do modelo e de visualização. O padrão MVC garante a independência entre dados e visualização e permite a flexibilização na escolha da plataforma gráfica. Assim, o mesmo **Modelo** (banco de dados) pode servir para aplicações desenvolvidas em diferentes plataformas gráficas.

O banco de dados foi projetado para abstrair as informações oriundas das diversas etapas do processo de fabricação de torres metálicas. Outras informações essenciais para a fase de detalhamento do processo também são armazenados no banco de dados. Os dados armazenados no banco de dados podem realimentar as etapas do processo ou gerar a representação gráfica do detalhamento utilizando a plataforma gráfica escolhida.

Os comandos personalizados criados pelo aplicativo CAD atuam de maneira idêntica aos comandos nativos do AutoCAD e podem ser acionados pela linha de comando, barra de ferramenta ou através do menu do AutoCAD. O sistema é ainda capaz de criar outros subprodutos tais como: croquis para fabricação e arquivos de dados para máquinas CNC com controle numérico.

Palavras Chave: Torres Metálicas, Detalhamento, Automação, Tecnologia CAD, API .NET, Banco de Dados.

ABSTRACT

This work presents the design and implementation of a computer system denominated TowerCAD for automation of the detailing process of steel towers structures by integrating CAD (Computer Aided Design), CAE (Computer Aided Engineering) and CAM (Computer Aided Manufacturing) technologies. The TowerCAD system integrates a CAD application and a database.

The system, called TowerCAD, is implemented using CAD technology, Object Oriented Programming paradigm (OOP) and Database technologies. The AutoCAD .NET technology, developed to be an AutoCAD programming interface (API), was used to customize AutoCAD and also was applied to develop interfaces for importing and exporting data, for creating commands, for data management and manipulation in the database . The database is accessed by the application through the ADO.NET classes using the standard SQL language for database and the programming language C #. AutoCAD was used as a platform for graphical representation.

An important contribution of this work is the application of concepts and fundaments of the programming architectural standard called MVC (Model-View-Control) in the development of the TowerCAD System. The three layers architecture of MVC to construct user's interfaces allows separation among information, visualization and control

The **Model** component represents the data and the respective rules for access and manipulation of the database. The **View** component, in this case represented by the graphical platform AutoCAD, executes the graphical visualization fo the data according to the user request. The **Control** interprets the users actions altering or not the state of the **Model** and of the **View**. **MVC** standard assures independence among control, data and visualization, which allows flexibility int the choice of the graphical platform. Therefore, the same Model (database) can be used in applications developed for different graphical platforms.

The database has been design to abstract information obtained from various phases of the manufacturing process of steel towers. Other essential information for the detailing phase are also stored in the database. The data stored in the database can feed back other phases of the process or generate the graphical representation of the detailing using an appropriate graphical platform.

The customized comands criated for the TowerCAD application behave in identical manner as the native AutoCAd commands and they can also be triggered by command line,tool bar or pull down menu as in AutoCAD. Besides detailing steel tower components, the system is also able to provide other sub-products such as manufacturing croquis and data files for numerical control CNC machines for fabrication.

Keywords: Steel Towers, Detailing, Automation, CAD Technology, API.NET and Database

LISTA DE ABREVIATURAS, SIGLAS E DEFINIÇÕES

ADN – Autodesk Developer Network - Rede mundial de desenvolvedores de aplicativos da Autodesk.

API – Aplication Program Interface – Um conjunto de bibliotecas previamente desenvolvidas para auxiliar durante o processo de criação de um aplicativo.

ARX – AutoCAD Runtime Extention.

ASCII – Arquivo baseado em um conjunto determinado de caracteres (Tabela ASCII).

Barras de Ferramenta (*Toolbars*) – Botões que representam atalhos dos comandos do aplicativo.

Bibliotecas – São conjuntos de funções e classes agrupadas que fornecem uma base de programação.

CAD – Computer Aided Design – Desenho auxiliado por computador.

CADTEC – Centro de Apoio, Desenvolvimento Tecnológico e Ensino da Computação Gráfica.

CAE - Computer-aided engineering – Engenharia auxiliada por computador.

CAM - Computer-aided manufacturing – Manufatura auxiliada por computador.

Entidade – São todos os objetos que possuem representação gráfica no AutoCAD como, por exemplo, linhas, círculos e textos.

Framework - É um conjunto de classes cooperantes que constroem um projeto reutilizável para uma específica classe de softwares. Classes podem ser customizadas através da criação de subclasses específicas para cada aplicação.

Namespace – No ambiente *Microsoft .Net, Namespace* é um contêiner que fornece contexto para os objetos que armazena

POO – Programação Orientada a Objetos.

SUMÁRIO

1. INTRODUÇÃO 13
1.1 Considerações Gerais 13
1.2 Motivações e Objetivos 14
1.3 Torres Metálicas 15
1.3.1 Famílias de Torres 17
1.4 CR Gontijo Engenharia de Projetos 18
1.5 Conteúdo por Capítulo 19
2. RECURSOS UTILIZADOS NO PROJETO 20
2.1 AutoCAD 21
2.1.1 Banco de Dados do AutoCAD 22
2.2 Programação Orientada a Objetos (POO) 23
2.3 ObjectARX 23
2.4 Framework .NET 24
2.4.1 Linguagem C# 27
2.4.2 Sistema "*Windows Forms*" 28
2.4.3 Como Lidar com Exceções em C# 30
2.4.4 Variáveis Estáticas 32
2.4.5 ADO.NET 33
2.5 API .NET para o AutoCAD 40
2.5.1 Componentes da API .NET para o AutoCAD 40
2.5.2 Qualidades 41
2.5.3 Estrutura de um Aplicativo .NET programa 41
2.5.4 Acesso ao Banco de Dados do AutoCAD 43
2.5.5 Acesso ao Banco de Dados externo 44
2.6 Banco de Dados 47
2.6.1 O Modelo Relacional de Dados 48
2.6.2 Linguagem SQL 51
3. O PROJETO DE TORRES 54
3.1 CADTEC – Centro Avançado de Desenvolvimento Tecnológico e Ensino da Computação gráfica 54
3.2 O Projeto de Torres Metálicas 55
3.2.1 O Processo Produtivo 55
3.2.2 Integração entre as etapas do processo 60
3.3 Etapas de Desenvolvimento 62
3.3.1 Levantamento Bibliográfico 62
3.3.2 Qualificação Computacional 62
3.3.3 Qualificação Técnica 63
3.3.4 Desenvolvimento do Sistema CAD 63
3.4 Desenvolvimento do Sistema TowerCAD 63
3.4.1 Desenvolvimentos Anteriores 63
3.4.2 Visão Geral 64
3.4.3 Características 67
4. O APLICATIVO CAD PARA DETALHAMENTO DE TORRES 70
4.1 Visão Geral 71

4.2 Template Utilizado 71
4.2.1 Layers utilizados 72
4.2.2 Blocos utilizados 73
4.3 Estrutura Global do Aplicativo 74
4.3.1 Classe `Initialization` 76
4.3.2 Classe `CdTcComands` 77
4.3.3 Classe `CdTcAuxiliar` 77
4.3.4 Classe `CdTcBancoDados` 83
4.3.5 Classe `CdTcBarra` 84
4.3.6 Classe `CdTcPrompt` 85
4.3.7 Classe `CdTcOutros` 86
4.3.8 Classe `CdTcDadosBarra` 87
4.3.9 Classe `CdTcDadosGeometria` 88
4.3.10 Classe `CdTcNormaParafuso` 88
4.3.11 Classe `CdTcIncluirNormaParafuso` 89
4.3.12 Classe `CdTcIncluirDistanciaParafuso` 90
4.3.13 Classe `CdTcIncluirComprimentoParafuso` 91
4.4 Comandos desenvolvidos 92
4.4.1 Comando `CBarra` 92
4.4.2 Comando `CPlaca` 94
4.4.3 Comando `CGeometria` 94
4.4.4 Comando `CCroqui` 95
4.4.5 Comando `CPe` 95
4.4.6 Comando `PParafuso` 97
4.4.7 Comando `LerArquivo` 99
4.4.8 Comando `GerarArquivo` 101
5. O BANCO DE DADOS 102
5.1 Visão Geral 102
5.2 A Construção da Estrutura do Banco de Dados 105
5.3 A Elaboração das Tabelas 106
5.3.1 Tabela `Torre` 108
5.3.2 Tabela `Barra` 110
5.3.3 Tabela `Emenda` 112
5.3.4 Tabela `Ligação` 113
5.3.5 Tabela `Chapa` 114
5.3.6 Tabela `Furo` 116
5.3.7 Tabela `Perfil` 116
5.3.8 Tabela `No` 118
5.3.9 Tabela `Tipo_Torre` 118
5.3.10 Tabelas sobre parafusos 119
5.3.11 Tabelas sobre o desenho 121
5.4 O Relacionamento entre as tabelas 123
5.4.1 Relacionamentos da tabela `Torre` 123

5.4.2 Relacionamentos da tabela `Barra` 125
5.4.3 Outros relacionamentos 126
5.5 Povoamento do Banco de dados 127
6. CONSIDERAÇÕES FINAIS 128
6.1 O Trabalho Desenvolvido e suas Contribuições 128
6.1.1 Contribuições Gerais 129
6.1.2 Contribuições Específicas para a Automação de Processos Produtivos 130
6.1.3 Contribuições Específicas para a Indústria de Torres Metálicas 130
6.2 Limitações do Aplicativo e Propostas para Trabalhos Futuros 131
6.2.1 Detalhamento da estrutura 131
6.2.2 Ligações 132
6.2.3 Emendas 132
6.2.4 Informações fornecidas pela projetista 132
6.2.5 Interferência entre elementos 132
6.2.6 Detalhamento 3D da estrutura 133
6.2.7 Lista de Materiais 133
7. REFERÊNCIAS BIBLIOGRÁFICAS 134

ÍNDICE DE FIGURAS

Figura 1-1 - Tipos de Torres 17
Figura 1-2 - Subestruturas das Torres Delta e Tronco Piramidal 18
Figura 2-1 - Estrutura do Banco de Dados do AutoCAD 22
Figura 2-2 - NET em contexto. Extraido de Microsoft (2009) 26
Figura 2-3 - Mecanismo multilinguagem permitido pelo Framework .NET. Extraído de Microsoft (2009) 27
Figura 2-4 – Arquivo de código da classe Windows Form 29
Figura 2-5 - Arquivo de interface da classe Windows Form 30
Figura 2-6 - Tratamento de excessões 32
Figura 2-7 - Arquitetura ADO.NET. Extraída de Microsoft (2009). 34
Figura 2-8 - Objeto DataView 36
Figura 2-9 - Campos e Registros 49
Figura 2-10 - Relacionamento Um para Vários 50
Figura 2-11 - Relacionamento tabela Furo x Barra 51
Figura 3-1 - Fluxograma do Processo 55
Figura 3-2 - Projeto básico da estrutura 56
Figura 3-3 - Detalhamento de um pé de uma torre 59
Figura 3-4 - Perfis prontos para serem montados. Extraída de Silva, J.B.G.F & all (2009) 60
Figura 3-5 - Composição do Sistema TowerCAD 65
Figura 3-6 - Fluxograma do Processo 66
Figura 3-7 - Interfaces com o projetista 67
Figura 3-8 - Representação gráfica x Dados 69
Figura 4-1 - Falta Layer 72
Figura 4-2 - Layers do Template 73
Figura 4-3 - Bloco dos parafusos 74
Figura 4-4 - Outros blocos 74
Figura 4-5 - Solution Explorer 75
Figura 4-6 - ClassView 76
Figura 4-7 - Classe `CdTcAuxiliar` 77
Figura 4-8 – Dados de entrada do método CriaGeometria 78
Figura 4-9 - Geometria da Torre 79
Figura 4-10 - Falta Layer 79
Figura 4-11 – Stub 80
Figura 4-12 - Texto da Barra 81
Figura 4-13 - Bloco não encontrado 82
Figura 4-14 - Bloco Posição 82
Figura 4-15 - Linha de comandos do AutoCAD 85
Figura 4-16 – Classe `CdTcDadosBarra` 87
Figura 4-17 - Classe `CdTcDadosGeometria` 88
Figura 4-18 - Classe `CdTcNormaParafuso` 89
Figura 4-19 - Classe `CdTcIncluirNormaParafuso` 90
Figura 4-20 - Classe `CdTcIncluirDistanciaParafuso` 91
Figura 4-21 - Classe `CdTcIncluirComprimentoParafuso` 91

Figura 4-22 - Ambiente do AutoCAD customizado pelo TowerCAD 92
Figura 4-23 - O Comando CBarra 93
Figura 4-24 - O Comando CPlaca 94
Figura 4-25 - Representação da geometria nos desenhos de detalhamento 95
Figura 4-26 - Croqui de uma peça para fabricação 95
Figura 4-27 - Detalhamento completo de um pé de uma torre 96
4-28 - Desenho de detalhamento de um pé gerado automaticamente pelo CPe 97
Figura 4-29 - Escolha da Norma de Parafusos 98
Figura 4-30 - Inclusão de Norma 99
Figura 4-31 - Seleção do arquivo para o povoamento do banco de dados 100
Figura 4-32 – Configuração de um arquivo .DAT modelo 100
Figura 4-33 – Formatação de um arquivo CAM para controle numérico gerado pelo comando GerarArquivo 101
Figura 5-1 - Parte da estrutura do Banco de Dados 104
Figura 5-2 - Elementos do detalhamento 107
Figura 5-3 - Tabela Torre 108
Figura 5-4 - Detalhe de dobra na cintura 109
Figura 5-5 - Tabela Barra 110
Figura 5-6 - Tabela Emenda 112
Figura 5-7 - Detalhe da emenda. Extraída de Silva, J.B.G.F & all (2009) 113
Figura 5-8 - Tabela Ligação 113
Figura 5-9- Ligação. Extraída de Silva, J.B.G.F & all (2009) 114
Figura 5-10 - Tabela Chapa 115
Figura 5-11 - Representação da Chapa 116
Figura 5-12 - Tabela Furo 116
Figura 5-13 - Tabela Perfil 117
Figura 5-14 - Tabela Perfil preenchida 117
Figura 5-15 - Tabela No 118
Figura 5-16 - Tabela Tipo_Torre 118
Figura 5-17 - Tabela Distancia_parafuso 119
Figura 5-18 - Distâncias mínimas 120
Figura 5-19 - Tabela Norma_parafuso 120
Figura 5-20 - Tabela Desenho 121
Figura 5-21 - Tabela Formato 121
Figura 5-22- Tabela Nota 122
Figura 5-23 - Notas 122
Figura 5-24 - Tabela Legenda 123
Figura 5-25 - Relacionamentos da tabela Torre 124
Figura 5-26 - Relacionamentos da tabela Barra 125
Figura 5-27 - Relacionamentos da tabela Desenho 127

1. INTRODUÇÃO

Este capítulo apresenta considerações gerais sobre o contexto do estado da arte da engenharia de torres, que de várias formas contribuiu para a decisão de desenvolvimento do presente trabalho. Em seguida, são apresentados a motivação e os objetivos da realização dessa dissertação. Uma breve leitura sobre as torres metálicas, o objeto de estudo do trabalho, é então apresentada. Finalmente, é feita uma rápida apresentação da empresa parceira e da forma de organização da dissertação em capítulos.

1.1 Considerações Gerais

Em uma época de crescimento econômico como a atual, são necessários grandes investimentos em infraestrutura para que o crescimento seja sustentável. Durante esse evento do crescimento os profissionais da área de engenharia ficam em evidência e são cada vez mais requeridos pelo mercado.

Os projetos têm prazos de execução cada vez mais apertados e as exigências de qualidade e produtividade definem prazos de execução mais curtos com menor quantidade de erros.

A execução de tarefas repetitivas manualmente pode ser considerada como uma das principais causas da existência de grandes quantidades de erros e da demora na execução de tarefas. O paradoxo da necessidade de executar tarefas mais rapidamente com menor probabilidade de erros se coloca como um dos maiores desafios para a engenharia moderna.

O surgimento dos computadores pode ser considerado um marco fundamental para o binômio agilidade e minimização de erros na execução de tarefas repetitivas. A velocidade da evolução das tecnologias dos computadores permitiu avanço equivalente, senão maior, da engenharia de "software" para a execução cálculos mais complexos e para a automação de processos, que despendiam muito tempo e geravam muitos erros.

A engenharia estrutural e outras ciências passaram por uma evolução importante com a automação de cálculos matemáticos através da Engenharia Auxiliada por Computador denominada tecnologia CAE (*Computer Aided Engineering*).

Posteriormente a tecnologia CAD (*Computer Aided Design*) ou Projeto Auxiliado por Computador permitiu a costumização e automação de processos de modelagem, visualização e detalhamento de produtos de uma forma contundentemente mais rápida, mais precisa e mais correta.

Em algumas áreas da engenharia estrutural, no entanto, o desenvolvimento da tecnologia CAD ainda não atingiu objetivos específicos citados acima. O processo de detalhamento de torres metálicas para a fabricação de pode ser considerado um exemplo. É um processo manual e que ainda requer bastante tempo para ser concluído.

1.2 Motivações e Objetivos

Um dos grandes problemas do desenvolvimento de projetos de estruturas de torres metálicas é a falta de automação do processo decorrente da grande carência de aplicativos capazes de atender as várias etapas do processo. Em especial, pode-se destacar a falta de automação da etapa de detalhamento, que constitui o gargalo de todo processo de fabricação.

Os softwares para detalhamento de estruturas metálicas existentes no mercado têm limitações, que dificultam sua utilização para atender as peculiaridades do detalhamento de torres metálicas. Eles não consideram a forma de detalhamento usual das empresas do mercado. As torres metálicas são estruturas especiais que, devido a sua geometria, a quantidade de planos existentes e ao tamanho de seus perfis, possuem uma grande quantidade de detalhes. Essa variedade considerável de detalhes, especialmente nas ligações, dificultam a automação da fase de detalhamento.

No caso das torres metálicas, o detalhamento se torna o gargalo de todo o processo produtivo, pois exige uma complexa análise sobre os resultados do dimensionamento

visando à transferência de dados do diagrama unifilar (ou silhueta) gerado pelo cálculo, para as reais dimensões envolvidas na montagem das torres em campo.

O processo de detalhamento de torres metálicas é, ainda hoje, bastante moroso e artesanal. Consiste ainda em uma importante fonte de erros, que pode gerar grandes prejuízos, incluindo a paralisação da obra. As plataformas gráficas ainda são utilizadas, na maioria das vezes, como pranchetas eletrônicas para desenhar o detalhamento.

Enquanto as etapas de cálculo e dimensionamento são concluídas em questão de dias e estão razoavelmente automatizadas, a etapa de detalhamento pode durar até meses. Em alguns casos, projetos de torres antigas são reaproveitados, mesmo a um custo maior, com o objetivo de cumprir os prazos que são cada vez mais apertados.

A falta de produtividade decorre também da ausência de sistemas CAD, CAE e CAM que permitam uma maior automação do seu processo produtivo, englobando todas as etapas do processo produtivo: entrada de dados ou modelagem, análise estrutural, dimensionamento, detalhamento e a fabricação.

Esta dissertação de mestrado endereça temas de pesquisa, que tem como objetivo global o desenvolvimento de um sistema para automação do processo de detalhamento das torres metálicas através da integração entre as tecnologias CAE (Computer Aided Engineering), CAD (Computer Aided Design) e CAM (Computer Aided Manufacturing).

Este trabalho tem como objetivo específico contribuir, de forma inovadora, com estudos para implementação de um sistema computacional (TowerCAD) segundo padrões de engenharia de "software" aplicados ao processo produtivo de estruturas de torres metálicas. Os resultados obtidos são apresentados através do sistema TowerCAD desenvolvido utilizando conceitos do padrão de arquitetura de software MVC (Modelo-Visualização-Controle), segundo Leite(2007), para integrar as etapas desse processo, porém garantindo a separação entre a informação, controle e representação gráfica.

Os objetos de estudo dessa dissertação visam desenvolver um sistema gerenciador de dados com a construção de um banco de dados específico, de um sistema de controle, que permite a transferência de dados e integração entre os módulos e de um aplicativo CAD para a visualização gráfica dos resultados. A proposta do TowerCAD é aumentar a produtividade do setor, minimizar erros e melhorar qualidade do processo e do produto, colocando assim as empresas em outro patamar de competitividade.

1.3 Torres Metálicas

As estruturas de torres metálicas para transmissão são elementos de sustentação dos cabos de linhas elétricas de transmissão de energia. As linhas de transmissão têm como objetivo fazer a ligação entre a usina geradora de energia e os centros consumidores de energia.

As dimensões e formas das torres metálicas dependem de diversos fatores, dentre os quais:

- O nível de tensão da linha. As linhas são classificadas como de alta tensão (<= 230 kV), extra alta tensão (230 kV a 700 kV) e ultra alta tensão (>700 kV). Quanto maior for a distância a ser percorrida maior será a tensão da linha.
- Distâncias elétricas estabelecidas.
- Número de circuitos.
- Obstáculos existentes na linha como, por exemplo, a travessia de rios.

As estruturas de torres metálicas treliçadas seguem uma arquitetura quase padronizada, sendo utilizadas para o transporte de energia em um ou dois circuitos. Para linhas de transmissão com grandes extensões e, por conseqüência, grandes tensões (> 69 kV), as torres metálicas são a solução mais econômica. Para tensões mais baixas (<69 kV), no entanto, outros materiais também são utilizados como, por exemplo, o concreto armado e a madeira.

O projeto arquitetônico da torre é feito pelo engenheiro calculista com base nos carregamentos, alturas a serem consideradas e distâncias de segurança. O calculista determina as aberturas superiores e inferiores das faces transversal e longitudinal e define o tipo de treliçamento principal e secundário a ser adotado. A partir dessas premissas são executados os cálculos de esforços solicitantes atuantes na estrutura e o dimensionamento das barras, das ligações e das fundações.

A estrutura deve ser analisada juntamente com a fundação, pois, a torre mais leve pode não garantir a solução global mais econômica.

Pode-se visualizar na Figura 1-1 alguns dos tipos de torres metálicas mais utilizadas, são eles:

- As torres auto-portantes do tipo Tronco Piramidal – Utilizadas em praticamente todos os níveis de tensões e em um e dois circuitos.
- As torres tipo Delta – Podem ser ainda do formato Cara de Gato.
- As torres Estaiadas.
- As torres Cross Rope.

As duas últimas formas de estrutura, referenciadas acima, foram projetadas ao longo do tempo, devido à necessidade de aumento da distância entre o ponto gerador e o ponto consumidor de energia. As duas primeiras são os tipos de torres de utilização mais freqüente em linhas de transmissão.

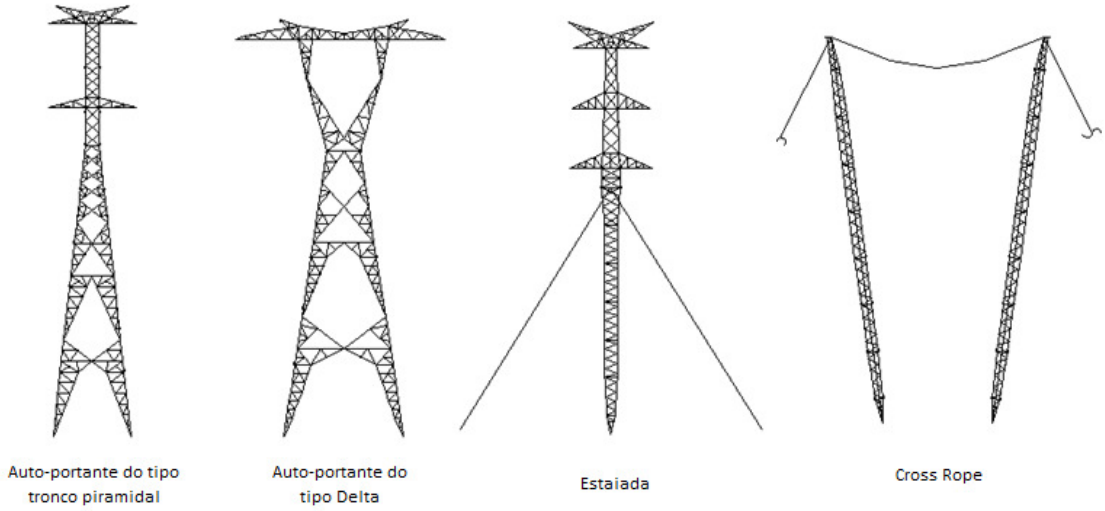


Figura 1-1 - Tipos de Torres

1.3.1 Famílias de Torres

Segundo Gontijo (1994), as torres metálicas são padronizadas em Famílias de Torres para o projeto e para a fabricação por questões de economia. Uma família de Torres é um conjunto de torres projetadas e montadas em uma linha de transmissão a partir de subestruturas idênticas ou similares. As famílias de Torres são compostas pelas seguintes subestruturas:

- Corpo básico – Comum em todas as torres da família. É constituído pela cabeça da torre e ainda pelo tronco básico inferior.
- Extensões do corpo básico – Sua presença depende da altura de uma torre específica na linha de transmissão que indica a altura da torre a ser montada. Normalmente é constituído por trechos com alturas variando de seis em seis metros.
- Pernas – Parte inferior da torre. Uma torre com seção quadrada ou retangular possui quatro pernas que podem ter tamanhos diferentes dependendo da topografia do local onde ela for instalada. A altura das pernas pode variar de 1,5 a 9,0m em segmentos de 1,5 de comprimento. Serve para evitar o nivelamento do terreno.

As estruturas são montadas em locais pré-determinados da linha, utilizando a combinação das subestruturas, visando atingir as necessidades de altura da torre e das condições topográficas. Conforme já dito, as torres não são projetadas uma a uma porque isso elevaria muito o custo de projeto e de fabricação já que seria necessário um projeto específico para cada locação de torre na linha de transmissão.

As subestruturas e os componentes das torres tipo Delta e Tronco Piramidal podem ser visualizados na Figura 1-2. Nas torres do tipo Tronco Piramidal, as subestruturas que compõem o corpo básico são os suportes dos cabos pára-raios, as mísulas, tronco da cabeça e o tronco básico inferior. Já nas torres do tipo Delta, o corpo básico é constituído pelas subestruturas: suportes dos cabos pára-raios, mísulas, viga, gambieta, delta e o tronco

básico inferior. As extensões e as pernas dos dois tipos de torres também podem ser visualizadas.

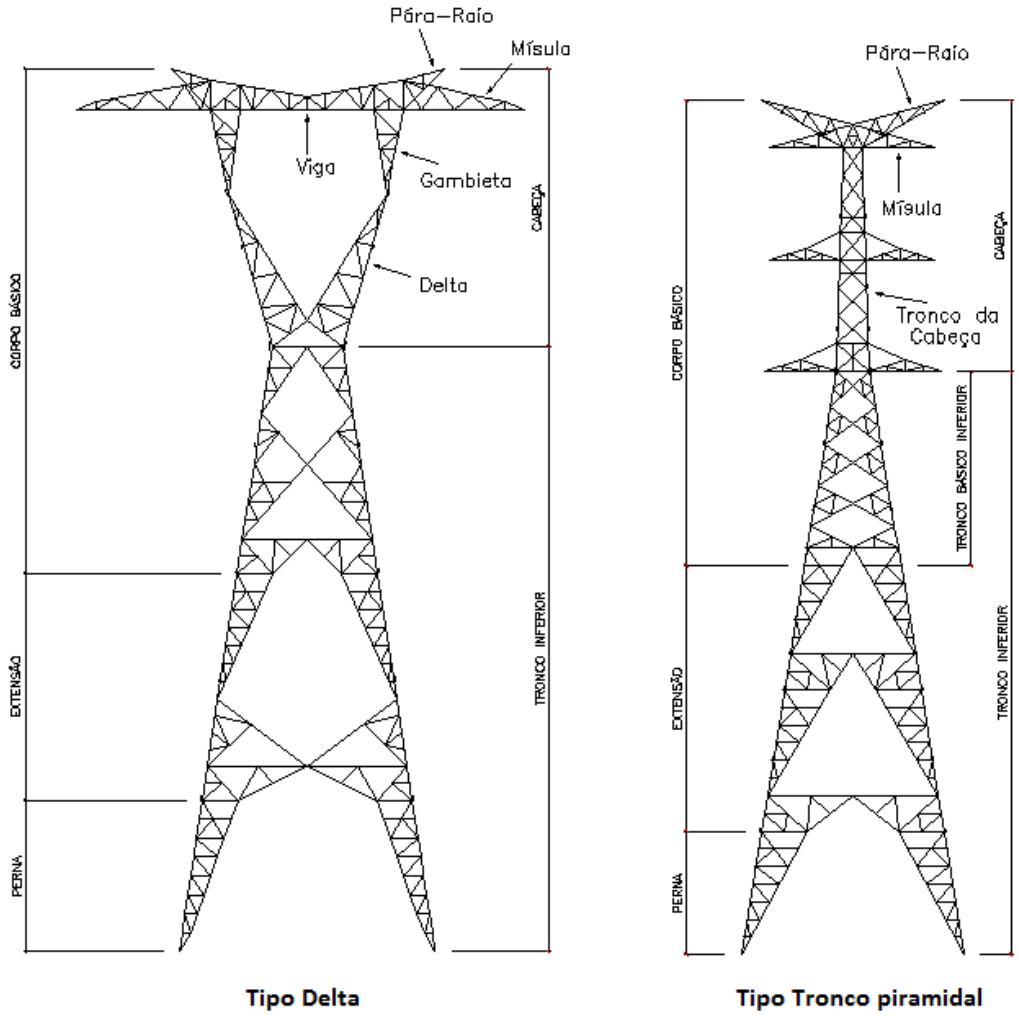


Figura 1-2 - Subestruturas das Torres Delta e Tronco Piramidal

1.4 CR Gontijo Engenharia de Projetos

Este trabalho foi desenvolvido em parceria com a empresa CR Gontijo Engenharia de Projetos, especializada em projeto de torres metálicas.

A equipe técnica da empresa participou e contribuiu significamente com sua experiência em projetos de torres metálicas. Os passos necessários para o desenvolvimento do aplicativo foram definidos pelos orientadores com o auxílio de profissionais da empresa parceira.

Os aspectos práticos dessa pesquisa e os estudos realizados relacionados ao projeto de torres, desde a concepção do projeto até o detalhamento, tiveram também uma contribuição importante da experiência dos profissionais da empresa C.R. Gontijo.

Os resultados práticos alcançados, pelo sistema TowerCAD, já estão sendo utilizados e testados pelos profissionais da empresa. Atualizações e sugestões de melhoria serão analisadas e levadas em consideração durante o desenvolvimento e aperfeiçoamento do aplicativo.

A participação de profissionais da área propiciou um enfoque prático ao aplicativo desenvolvido.

1.5 Conteúdo por Capítulo

No presente capítulo definiu-se o contexto global do trabalho dissertando-se sobre aspectos gerais da pesquisa, sobre seus objetivos e justificativas, sobre torres metálicas e suas famílias e sobre a parceria com a empresa do setor. Uma breve introdução sobre engenharia de torres metálicas é feita com ênfase especial ao detalhamento de estruturas metálicas.

No segundo capítulo são apresentadas todas as ferramentas utilizadas no desenvolvimento do trabalho com uma breve revisão de seus fundamentos. O AutoCAD, o API .NET e as tecnologias para banco de dados são algumas das ferramentas mencionadas.

No terceiro capítulo é feito um breve relato sobre o Grupo de Pesquisa e sobre é o CADTEC – Centro Avançado de Desenvolvimento Tecnológico e Ensino da Computação gráfica, onde esta pesquisa foi inserida, em seguida são mostradas as etapas de desenvolvimento do trabalho, são descritas também as etapas do processo de desenvolvimento de um projeto de torres metálicas e finalmente são apresentadas as características importantes o sistema TowerCAD desenvolvido.

No quarto capítulo é apresentado o aplicativo CAD para detalhamento de torres desenvolvido para esse trabalho. Suas características, sua estrutura global com todas as classes criadas e todos os comandos criados para customizar o AutoCAD são detalhados nesse capítulo.

No quinto capítulo são apresentadas as caracterícas relevantes do Banco de Dados do sistema computacional desenvolvido. Sua independência da plataforma gráfica é ressaltada. Suas tabelas e os relacionamentos entre tabelas são detalhados.

No último capítulo são apresentadas as considerações finais com apresentação das contribuições geradas pelo trabalho, as limitações do aplicativo e ainda as propostas para trabalhos futuros.

2. RECURSOS UTILIZADOS NO PROJETO

Para o desenvolvimento deste projeto de pesquisa foram utilizadas como ferramentas de desenvolvimento o AutoCAD 2008, a API .NET que é baseada na tecnologia .NET, a linguagem padrão para banco de dados SQL, o Microsoft Access 2007 e o Microsoft Visual Studio 2005.

Foram ainda estudadas outras tecnologias que permitem desenvolver aplicativos específicos no AutoCAD e que foram utilizadas em trabalhos anteriores como a interface de programação de aplicativos ObjectARX e a biblioteca de classes MFC responsável pela criação de interfaces com o usuário.

O AutoCAD foi a plataforma gráfica escolhida para o desenvolvimento do aplicativo TowerCAD por ser uma plataforma de grande aceitação no mercado, amplamente utilizada pelos escritórios de empresas de engenharia civil, e por possuir uma arquitetura aberta para o desenvolvimento de aplicativos para áreas específicas através de interfaces de programação.

Esse é um trabalho pioneiro no programa de Pós-graduação em Engenharia de Estruturas da UFMG - PROPEEs na utilização da API baseada na tecnologia .NET para o desenvolvimento de sistemas CAD. Sua escolha foi baseada principalmente na facilidade

de interligação de banco de dados externos com a plataforma gráfica e em outras vantagens que serão descritas na seção 2.5.2. O aplicativo criado com auxílio da interface de programação de aplicativos (API) .NET, assim como os criados com a API ObjectARX, são vinculados dinamicamente ao AutoCAD e os comandos criados operam do mesmo modo que os comandos nativos.

O Microsoft Visual Studio é uma ferramenta desenvolvida pela Microsoft em um ambiente de desenvolvimento integrado de sistemas computacionais para o Windows. O Visual Studio foi escolhido como o "*framework*" para o desenvolvimento do aplicativo por permitir uma programação mais rápida e eficiente, devido às suas ferramentas internas de programação.

Os códigos e a estrutura de banco de dados desenvolvidos podem ser utilizados em qualquer sistema gerenciador de banco de dados com apenas pequenas modificações a serem feitas. Para o aplicativo criado, as funcionadas oferecidas pelo Microsoft Access foram suficientes. A linguagem SQL é padrão para programação de banco de dados e é utilizada independentemente do sistema gerenciador de banco de dados.

2.1 AutoCAD

O AutoCAD segundo Baldam(2007) é uma plataforma gráfica genérica desenvolvida pela empresa Autodesk. É a plataforma gráfica mais difundida do mundo nos escritórios de cálculo para projetos assistidos por computador. Possui uma série de recursos e comandos para a geração de entidades gráficas com elevada precisão geométrica.

O AutoCAD possui uma estrutura de banco de dados interna que permite o armazenamento de informações sobre objetos e entidades (objetos criados com representação gráfica) necessárias à manipulação das características geométricas e à construção de sua representação gráfica. Um desenho do AutoCAD é formado por um conjunto de objetos armazenados nesse banco de dados. O conhecimento dessa estrutura é fundamental para a programação do AutoCAD.

O AutoCAD contribuiu para uma grande revolução nas áreas de tecnologias CAD que aconteceu com a abertura da arquitetura do código para customização e personalização através do desenvolvimento de aplicativos CAD para áreas específicas.

Atualmente, o AutoCAD pode ser programado através da linguagem Autolisp (a partir da versão 12, com recursos limitados), pela API (Interface para programação de aplicativos) ObjectARX e utilizando a nova API baseado na tecnologia .NET.

2.1.1 Banco de Dados do AutoCAD

O banco de dados do AutoCAD armazena um conjunto de objetos gráficos e não gráficos que formam um arquivo do AutoCAD. Os objetos que possuem representação gráfica são denominados entidades e os objetos que não possuem representação gráfica são denominados objetos.

O banco de dados do AutoCAD é composto por uma estrutura constituída por tabelas de símbolos e pelo dicionário de objetos nomeados conforme ilustrado na Figura 2-1.

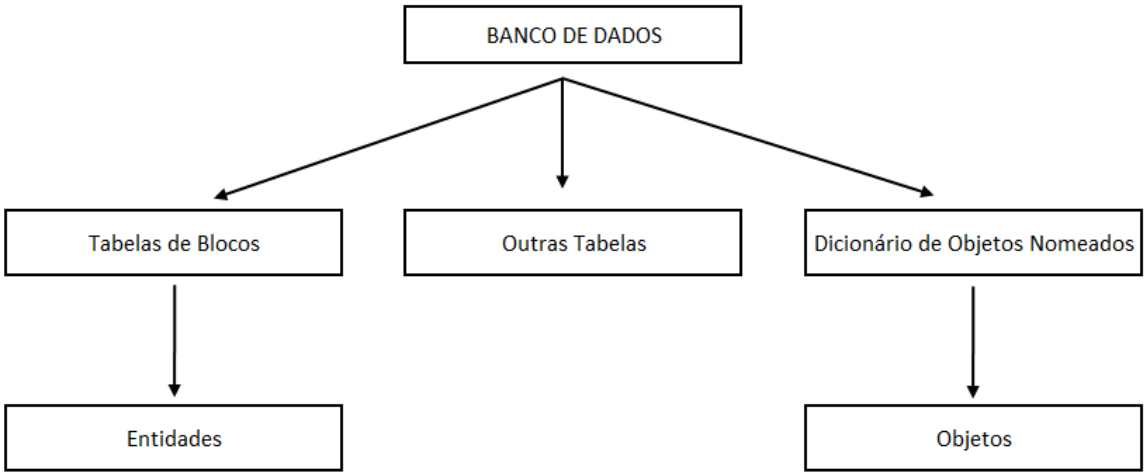


Figura 2-1 - Estrutura do Banco de Dados do AutoCAD

Existem nove tabelas de símbolo e apenas um dicionário de objetos nomeados. As tabelas de símbolo não podem ser criadas nem destruídas por um aplicativo. Apenas seu conteúdo pode ser modificado quando se adiciona ou altera os registros existentes. Cada tabela armazena um tipo específico de objeto como, por exemplo, entidades, layers, tipos de linhas e estilos de dimensionamento. A tabela de blocos armazena as entidades criadas.

As tabelas de símbolo e os dicionários são os dois tipos de objetos de armazenamento existentes no banco de dados do AutoCAD. Quando uma entidade ou objeto é inserido em um objeto de armazenamento, ele recebe um identificador único através do qual poderá ser acessado dentro do banco de dados do AutoCAD. Cada arquivo com a extensão .dwg criado no AutoCAD possui um único banco de dados.

Um identificador, ou ID, identifica e associa o objeto ao seu banco de dados, mesmo quando vários bancos de dados estão abertos ao mesmo tempo, ou seja, mesmo quando vários arquivos .dwg estão abertos ao mesmo tempo no *software* AutoCAD. Esse identificador não é, no entanto, necessariamente o mesmo quando um arquivo .dwg é fechado e depois reaberto. Para identificar o objeto de maneira única existe, outro identificador denominado "*handle*", que permanece associado ao mesmo objeto, mesmo quando o arquivo é fechado e aberto novamente.

Os objetos armazenados nas tabelas ou nos dicionários do AutoCAD podem ser encontrados pela utilização de iteradores. Iteradores são objetos responsáveis por percorrer as tabelas de símbolos e os dicionários pesquisando o seu conteúdo.

2.2 Programação Orientada a Objetos (POO)

Segundo Leite (2002), a Programação Orientada a Objetos constitui um padrão de programação, que possibilita um alto reaproveitamento de códigos, com sistemas formados a partir de sistemas menores ou projetos. Os sistemas menores ou projetos são constituídos por classes e objetos onde os dados e os procedimentos estão juntos em um só elemento.

Uma linguagem orientado a objetos possui uma série de mecanismos e conceitos básicos. Dentre os mecanismos pode-se citar: os objetos, as classes, o membros das classes, os tipos de acesso, as mensagens, as heranças etc. Alguns dos conceitos básicos que podem ser citados são: abstração, encapsulamento, polimorfismo, modularidade, persistência e a tipificação.

A Programação Orientada a Objetos se difere da Programação Estrutural pela maneira através da qual os dados e procedimentos se intercomunicam. Nos sistemas estruturados, a implementação é feita através de estruturas, que englobam procedimentos que se relacionam pela troca de dados. Na análise orientada a objetos, dados e procedimentos estão agrupados formando as classes de objetos. O funcionamento se dá através do relacionamento e troca de mensagens entre os objetos.

Duas das grandes vantagens da programação orientada a objetos são o alto índice de re-utilização de código e a maior facilidade de manutenção do sistema. Podemos destacar ainda algumas características modulares interessantes para desenvolvimento de s*oftware* como a portabilidade, reusabilidade, confiabilidade e alguns conceitos já citados como a abstração, o encapsulamento, a herança e o polimorfismo.

A programação orientada a objetos é ideal para aplicativos com grande grau de elaboração e elevado potencial de desenvolvimento e aprimoramento. O paradigma da POO foi utilizado na elaboração desse trabalho.

2.3 ObjectARX

ObjectARX segundo Malard(2009) é uma API (Interface para Programação de Aplicativos), que utiliza a linguagem C++ para personalizar a plataforma gráfica AutoCAD. O AutoCAD R14 foi a primeira versão aberta para o desenvolvimento de

aplicativos utilizando essa API. ObjectARX é a API mais completa disponível para a personalização do AutoCAD atualmente.

Baseado nos paradigmas da programação orientada a objetos, o ObjectARX permite derivar novas classes com a mesma funcionalidade de classes nativas do AutoCAD. Além disso, é possível acessar as estruturas de banco de dados do sistema gráfico do AutoCAD e ainda criar novos comandos. Para garantir o controle do AutoCAD sobre as entidades ou objetos personalizados é necessário definir através do polimorfismo um conjunto mínimo de funções a serem sobrecarregadas.

Usualmente, o projeto desenvolvido na API ObjectARX é dividido em dois módulos:

- ARX – Armazena as funções, comandos e classes derivadas do MFC, inicialização, carregamento e descarregamento do aplicativo
- DBX – Armazena classes definidas e utilizadas pelo aplicativo.

O aplicativo desenvolvido em ObjectARX fica integrado ao AutoCAD em forma de biblioteca de classes dinâmicas.

2.4 Framework .NET

Sistemas computacionais baseados na tecnologia .NET utilizam técnicas para incrementar o desempenho de programas interpretados em máquinas virtuais. A máquina virtual, funciona como um computador fictício implementado através de "*software"*, que executa programas isolada e eficientemente como um computador real. A principal caracteristica de máquinas virtuais como o Java e .NET é o processo de compilação intermediária, que traduz o código fonte para uma representação conhecida como *"bytecode"*. O *bytecode* não está vinculado a nenhum código de máquina ou sistema operacional e portanto pode ser transferido para várias arquiteturas de computador. A representação intermediária (*bytecode*) pode ser interpretada e/ou executada em tempo real na máquina virtual de cada arquitetura específica. Esta técnica, conhecida como JIT, compilador *"just-in-time"* converte em tempo de execução instruções do formato *bytecode* para código de máquina.

Nas arquiteturas, que oferecem ambientes com recursos de JIT, os "*bytecodes"* resultantes da compilação do programa fonte são executados pela máquina virtual, que realiza a conversão desse bytecode para código de máquina nativo enquanto o executa. Normalmente, trechos do código fonte são traduzidos em tempo de execução e por isto o nome dado *"just-in-time"*.

No caso do *Framework .NET* a compilação intermediária é executada pela linguagem MSIL (Microsoft Intermediate Language). Todo arquivo .NET é pré-compilado nesta linguagem intermediária e é compilado em tempo de execução na máquina do cliente através da Linguagem Comum em Tempo de Execução - CLR(Common Language Runtime).

Para escrever um código gerenciado é importante conhecer bem a CLR, que é um importante componente do *Framework .NET*. Código gerenciado é a definição aplicada a qualquer *"software*" que possa ser executado no *Framework .NET* e que vise a compilação intermediária através da CRL, enquanto o código que não passa por essa fase é conhecido como código não gerenciado. Uma das evoluções da tecnologia .NET é a independência do sistema operacional e outra que pode ser citada é a possibilidade do desenvolvedor escolher a linguagem que irá programar sem perdas de recursos ou performance. Isto inclue aplicativos em linguagens de alto nível como C#, além de outras linguagens, que tenham como alvo o *Framework .NET,* como o Java, C++, VB, etc. As linguagens de alto nível são linguagens com um nível de abstração elevado, longe do código máquina e mais próximo da linguagem humana.

Entender porque a CLR é necessária e como ela funciona significa entender que pode ser feito com essas linguagens. A plataforma *Framework .NET* foi desenhada para executar aplicações na Internet e a CRL é o ambiente de execução dessa plataforma fornecendo serviços fundamentais como gerenciamento de memória, gerenciamento de segmento e arquitetura de comunicação remota. As exigências da CRL para o desenvolvimento de uma aplicação podem ser itemizadas assim:

- **Execução binária segura** de componentes de "software" originados através da internet ou de redes sem o receio do sistema ser violado.
- **Desempenho –** Uma importante caracteristica de um **"***software***"** para Internet é que ele possa ser executado na linguagem de máquina nativa do sistema hospedeiro para aproveitar as vantagens do "*hardware*" no desenvolvimento da aplicação.
- **Redução de Bugs(erros)** Aplicações desenvolvidas para a Internet precisam ser robustas, mesmo que sejam executadas localmente na linguagem nativa da máquina do cliente. Aplicativos desenvolvidos com a CRL, que não tenham a Internet como alvo, mas que apresentam uma interconexão ou dependência (banco de dados) com "*software*" executado ou distribuído remotamente também precisam se robustos.
- **De Fácil Integração:** Outra exigência da CRL para o desenvolvimento de aplicações é que essas sejam capazes de integrar com outros "softwares" e possa rodar em várias paltaformas.

A CLR (*"Common Language Runtime")* preenche todos esses requisitos de uma forma ou de outra.

O contexto da plataforma .NET pode ser visualizado na Figura 2-2.

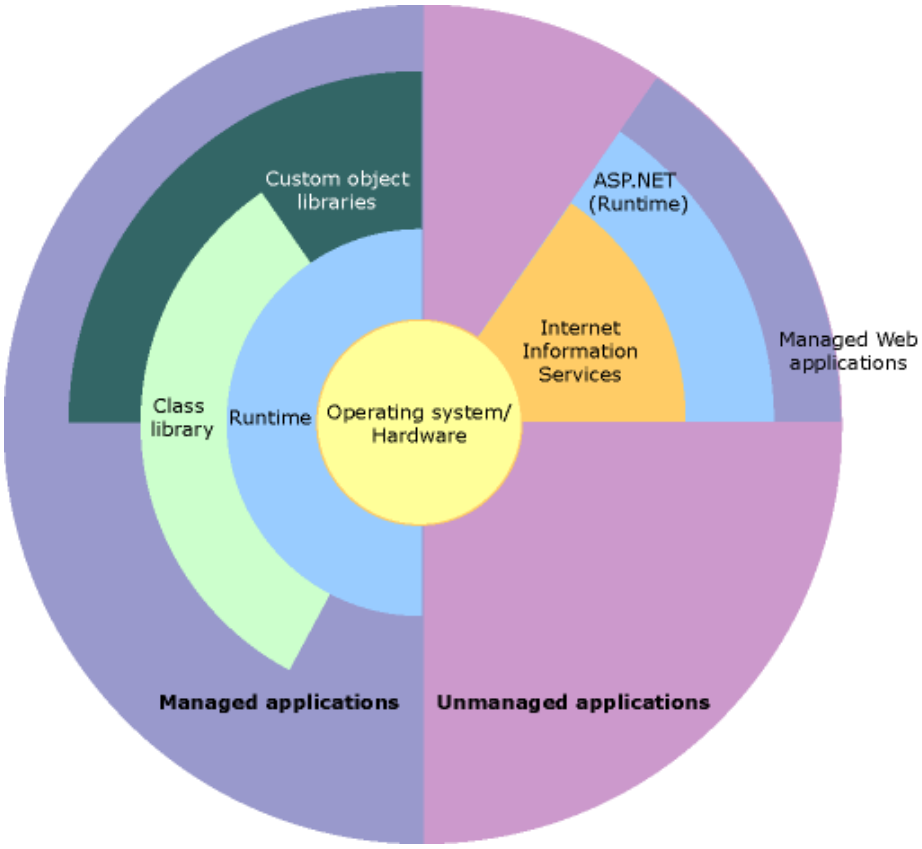


Figura 2-2 - NET em contexto. Extraido de Microsoft (2009)

Uma das evoluções da tecnologia .NET é a independência do sistema operacional e outra que pode ser citada é a possibilidade do desenvolvedor escolher a linguagem que irá programar sem perdas de recursos ou desempenho.

As bibliotecas de classes do *framework* estão organizadas em objetos denominados "*namespaces*" (contêiners que fornecem o contexto para os objetos que armazena) para facilitar a localização de classes e funções específicas. Essa bibliotecas têm código aberto, permitindo a utilização de objetos diretamente ou a especialização dos mesmos. O Framework é composto por bibliotecas ou "*assemblies"*(DLL ou EXE) que dão suporte a qualquer linguagem que rode dentro dele.

Aplicativos desenvolvidos com *Framework .NET* adotam a seguinte sequência; um programa é escrito em qualquer das mais de vinte linguagens de programação disponíveis, o código fonte gerado pelo programador é então compilado pela linguagem escolhida produzindo um código intermediário na linguagem MSIL (Microsoft Intermediate Language). O mecanismo multilinguagem pode ser visualizado na Figura 2-3.

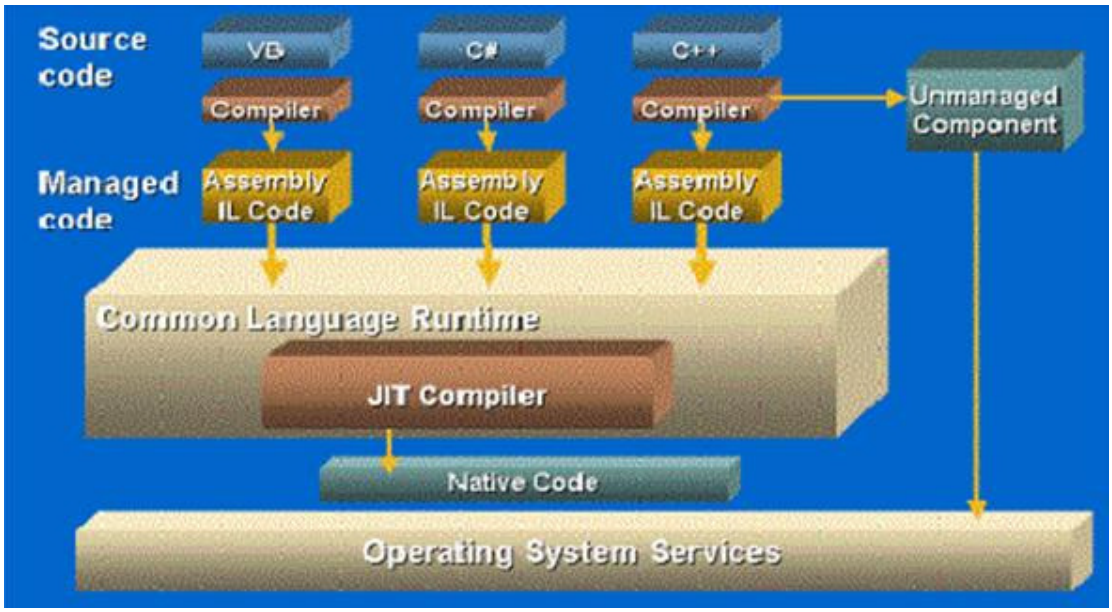


Figura 2-3 - Mecanismo multilinguagem permitido pelo Framework .NET. Extraído de Microsoft (2009)

Este novo código fonte gera um arquivo chamado de "*Assembly*", de acordo com os seguintes tipos de projeto:

- EXE - Arquivos Executáveis, Programas
- DLL - Biblioteca de Funções
- ASPX - Página Web
- ASMX - Web Service

2.4.1 Linguagem C#

C# segundo Sharp (2008) é uma linguagem de programação simples, moderna, orientada a objetos e segura. As raízes da linguagem C# estão na familia das linguagens C e é imediata a familiariada da mesma para programadores de C, C++ e Java. C# foi criada pela Microsoft dentro do empreendimento de desenvolvimento da tecnologia .NET segundo os padrões da ***ECMA-334 e ISO/IEC 23270***. Juntamente com o VB.NET é a linguagem mais importante da plataforma .NET.

Além de ser uma linguagem orientada a objetos, C# suporta também programação de componentes orientados a objetos. Projetos de "*software*" modernos dependem cada vez mais de componentes de aplicações em formato de pacotes de funcionalidades auto-contidas. Esses pacotes apresentam padrões de programação com propriedades, eventos e métodos, apresentam ainda atributos que fornecem informações declarativas sobre o componente e também incorporam sua própria documentação. C# fornece recursos de

linguagem para sustentar esses conceitos, que fazem do C# uma linguagem natural para a criação e o uso de componentes de "software".

Várias características da C# contribuem para a construção de "*softwares*" robustos e duráveis:

- **Coleta de Lixo** recupera automaticamente a memória utilizada por objetos ociosos
- **Manejo de Exceções**: fornece um controle estruturado e extensivo de deteção e recuperação de erros
- **Tipo seguro:** projeto da linguagem torna impossível a leitura de variáveis não iniciadas ou a indexação de vetores além do seu limite
- **Flexível** – Pode ser executado na máquina corrente ou em outra pela web.
- **Poderoso** – Essencialmente os mesmos comandos do C++.
- **Fácil de usar** – Modifica os comandos responsáveis por mais erros no C++ e gasta menos tempo com erros.
- **Visualmente orientado**.
- **Internet "friendly".**

O aplicativo resultado desse trabalho foi desenvolvido na linguagem C#.

2.4.2 Sistema "*Windows Forms"*

"System Windows Forms" é um ambiente do Windows, cuja biblioteca de classes foi incluída como parte do *Framework .NET.* para criação de interfaces gráficas de programação de aplicativos (API - Application Programming Interface). Através desse ambiente é possível acessar os elementos nativos de interface do MS Windows pelo empacotamento de APIs existentes do Windows em código gerenciado. As interfaces criadas e customizadas funcionam e aparentam para o usuário como se fossem interfaces nativas do Windows.

As interfaces com o usuário para entrada de dados do aplicativo desse trabalho, construídas para comunicar através de formulários, foram derivadas do ambiente "*System Windows Forms"*. Embora seja considerado por alguns como substituto do MFC (*Microsoft Foundation Class Library*), que é baseado no C++, o ambiente "*System Windows Forms"* não oferece um paradigma comparável ao MVC (Modelo-Visualização-Controle). Algumas bibliotecas terceirizadas foram criadas para suprir essa funcionalidade, das quais a mais utilizada tem sido a ***User Interface Process Application Block*** **(Bloco de Aplicação do Processo de Interface do Usuário)**, que pode ser baixada gratuitamente e inclui o código fonte para exemplos rápidos.

As classes derivadas do ambiente "*System Windows Forms"* possuem um arquivo, que armazena o código da classe e outro no qual são feitas as alterações do "*design*". Essas classes são do tipo que possui uma representação gráfica.

A classe "*Form*" (*System.Windows.Forms*) é uma parte fundamental daquele ambiente, por ser um bloco chave na construção de aplicativos Windows. Na Figura 2-4, pode ser visualizado o arquivo que armazena o código da classe costumizada `CdTcDadosBarra` (desenvolvido neste trabalho e descrita na seção 4.3.8), que é herdeira (derivada) da classe *System.Windows.Forms.Form*.

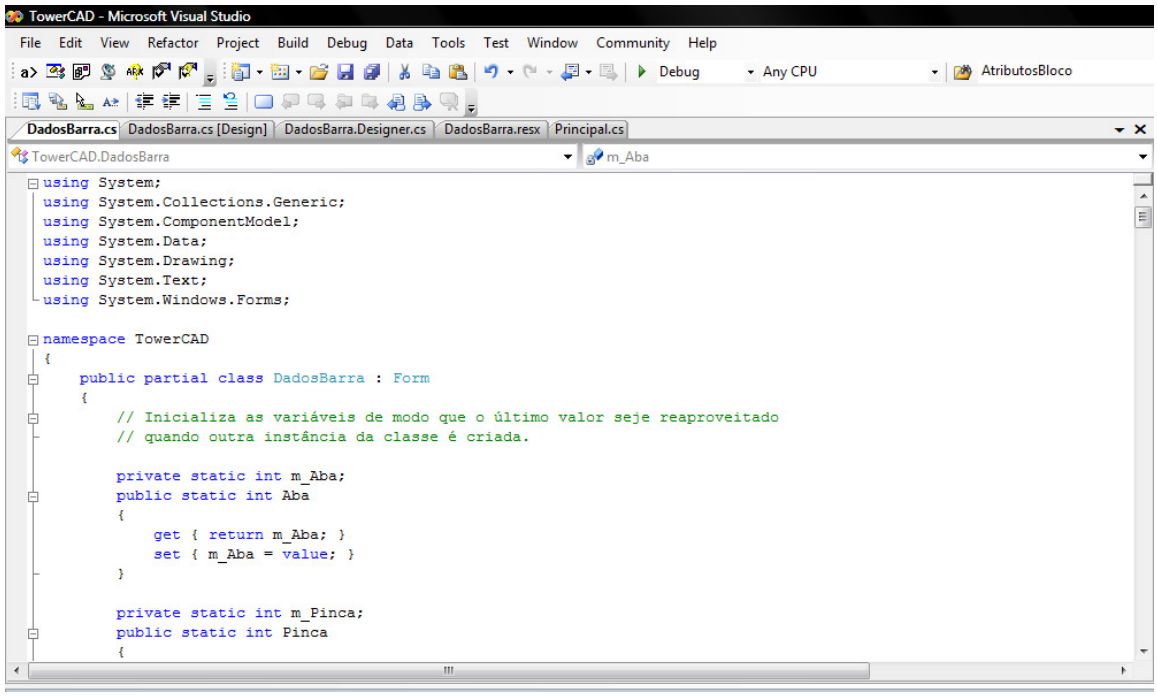


Figura 2-4 – Arquivo de código da classe Windows Form

O arquivo de código de uma classe derivada do "*System Windows Forms*" é bem semelhante a outros arquivos de código com exceção da primeira linha de declaração da classe, que informa que a classe é derivada.

O arquivo de representação gráfica da classe *"Form"*, no qual podem ser feitas as alterações no *"design"*, disponibiliza um quadro visual com botões, ícones e barras juntos na mesma janela e por isto é bem fácil de ser manipulado. Disponibiliza também uma barra de ferramentas com os vários controles que podem ser acrescentados à janela de interface. Os controles podem ser facilmente movidos dentro da janela criada de modo a tornar a interface mais agradável para o usuário. A Figura 2-5 permite a visualização do *"design"* da classe `CdTcDadosBarra` e a barra de ferramentas com os controles, que podem ser acrescentados.

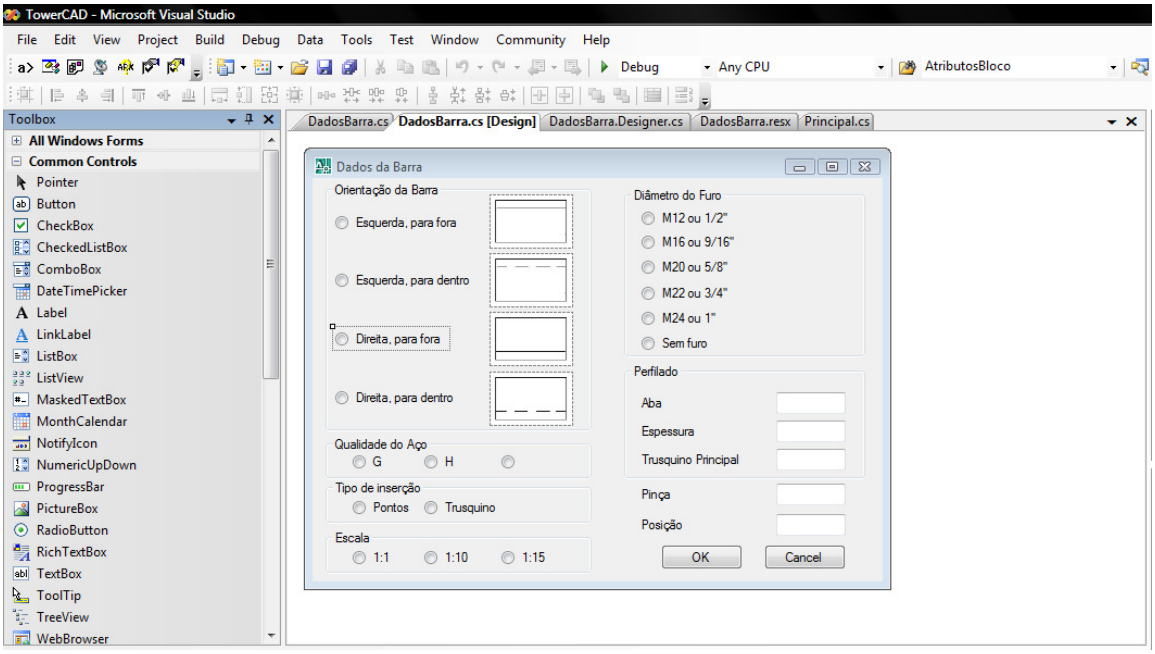


Figura 2-5 - Arquivo de interface da classe Windows Form

2.4.3 Como Lidar com Exceções em C#

Os recursos para lidar com exceções na linguagem C# permitem o programador controlar condições de erros ou situações de anormalidade de uma maneira estruturada e segura em nível do sistema operacional e do aplicativo. Desta forma o programador pode separar o fluxo normal do código da lógica para lidar como o erro.

Uma exceção pode representar uma variedade de condições anormais, como, por exemplo, o uso de uma referência nula de um objeto detectada pelo sistema em tempo de execução ou ainda uma sequência inválida de entrada *(input)* fornecida pelo usuário e detectada pelo código. Um código cria (*throw*) uma exceção quando detecta uma condição de erro e um código captura (*catch*) uma exceção quando consegue tratar e lidar com o erro. Em C# uma exceção é um objeto, que encapsula várias informações sobre o erro ocorrido, tal qual uma divisao por zero e a respectiva mensagem descritiva do erro. Os objetos de exceção são instanciações (exemplos) da classe mãe "*System.Exceptions*" ou de uma de suas classes filhas. Existem várias classes de exceção definidas no "*Framework .NET",* que são usados para vários objetivos. Programadores também podem definir suas próprias classes de exceção herdeiras da classe "*System.Exceptions*".

Na linguagem C# existem 3(tres) definições de códigos para lidar e tratar a exceção, que são:

1. *try/catch* – nesse bloco o código verifica a existência de exceção(*try*) e a cria se ela ocorrer (*throw)* e em seguida captura e trata a exceção (*catch*). Caso não ocorra nenhuma exceção, o bloco *catch* não é executado
2. *try/catch/finally* - nesse bloco o código verifica a existência de exceção(*try*), captura e trata a exceção (*catch*) se ela ocorrer e SEMPRE executa a declaração "*finally*" para especificar finalização de código independentemente da exceção ter ocorrido ou não. Este tipo de declaração é útil para as situações de fechamento de conexões, de arquivos ou mesmo liberar algum recurso caro para o sistema operacional. (não entendi)
3. *try/finally* – aqui o código verifica a existência de exceção(*try*), e SEMPRE executa a declaração "*finally*" para especificar finalização de código. Qualquer exceção que ocorrer será jogada para depois do "*finally*".

Um exemplo da utilização de exceções neste trabalho é a entrada de dados pela interface gráfica criada para o comando `CBarra` (criado neste trabalho e descrito na seção 4.4.1). Na caixa de texto, (*textBox),* chamada de *ButtonAba*, o usuário deve digitar o valor da aba da cantoneira. São aceitos apenas valores inteiros. Com a intenção de verificar se o usuário está utilizando o programa com a devida consistência , ou seja, se está digitando um inteiro e não algum outro tipo de valor, o programa checa o conteúdo da caixa de texto, ButtonAba toda vez que o usuário digita um caractere no teclado.. O código da checagem é exibido abaixo.

```
        private void ButtonAba_KeyUp(object sender, KeyEventArgs e)
        {
            try
            {
                Aba = Convert.ToInt16(this.ButtonAba.Text);
            }
            catch
            {
                MessageBox.Show("Digite um número.", "Valores
incorretos.", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
                ButtonAba.Text = "0";
            }
        }
```

O programa tenta converter o texto que está escrito na caixa de texto (*textBox)* em um inteiro. Caso isso não seja possível, uma exceção é lançada e uma caixa de mensagens aparece na tela informando ao usuário que o programa não está sendo utilizado corretamente. A caixa de texto (*textBox)* recebe então o valor zero (0) e aguarda que o usuário digite o valor corretamente. O resultado gráfico é mostrado na Figura 2-6 quando o usuário digita a letra s na caixa de texto ButtonAba.

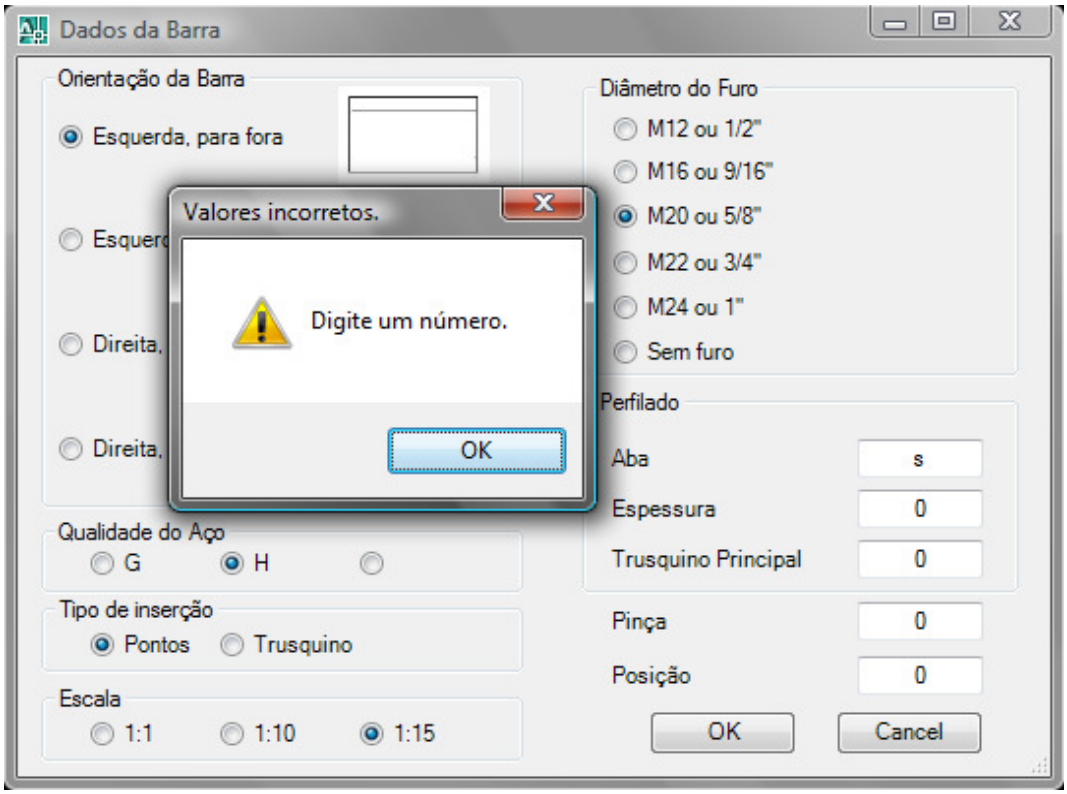


Figura 2-6 - Tratamento de excessões

2.4.4 Variáveis Estáticas

Na liguagem C e nas suas descendentes, o termo variável estática tem pelo menos três significados relacionados com a semântica da palavra chave *"static"* da linguagem. Aqui vamos nos referir apenas à variável estática local, que só pode ser acessada de dentro de um função ou procedimento onde foi declarada. A característica diferente da variável estática está na duração do armazenamento do valor alocada a ela pela função. O valor alocado durante uma chamada da função é preservado e estará disponível quando a funçao for novamente chamada. As variáveis estáticas são utilizadas para preservar os valores armazenados.

As variáveis estáticas são inicializadas somente uma vez dentro de uma função, quando o programa é disparado. Essas variáveis são automaticamente inicializadas com zero ou outro valor na alocação de memória exatamente como as variáveis externas. Diferentemente seus valores prévios continuam a existir e são retidos mesmo depois de sair da função.

Uma situação muito comum em programação é a conveniência de deixar valores digitados para a execução de um comando, disponíveis para a chamada seguinte desse comando. Isso aumenta a produtividade do comando, uma vez que o usuário não necessitará digitar

novamente todos os dados solicitados pelo comando e sim apenas os dados que serão modificados.

A variável estática no ambiente de Formulários do Sistema Windows (***"System Windows Forms")*** do Framework .NET funciona exatamente da mesma forma. Todas as vezes que um comando é evocado, um objeto da classe ambiente "*Windows Forms"* é criado. Ao ser criado, o objeto não possui seus atributos preenchidos. Os atributos são preenchidos na inicialização do objeto ou posteriormente.

Para que os dados possam ser recuperados, as variáveis estáticas são inicializadas na definição da classe "*Windows Forms"* e armazenam os valores na classe e não nos objetos criados. Assim ao inicializar o objeto, o mesmo pode então acessar os dados que estão armazenados na classe. Abaixo se encontra um exemplo de código de inicialização de uma variável estática na classe `CdTcDadosBarra`, que pertence ao programa ("*namespace"*) TowerCAD.

```
namespace TowerCAD
{
    public partial class DadosBarra : Form
    {
        // Inicializa as variáveis de modo que o último valor seja
        //reaproveitado quando outra instância da classe é criada.

        private static int m_Aba;
        public static int Aba
        {
            get { return m_Aba; }
            set { m_Aba = value; }
        }
    }
}
```

As variáveis estáticas foram utilizadas no aplicativo CAD principalmente para manter os dados de um comando armazenados na interface gráfica quando o comando é acionado outra vez pelo usuário.

2.4.5 ADO.NET

O "*Framework .NET"* é uma completa biblioteca de tipos reutilizáveis de objetos-orientados, que podem ser usadas para o desenvolvimento de aplicativos. A biblioteca de classe do "*Framework .NET"* inclue o ADO.NET", "ASP.Net" e "Windows Forms".

A tecnologia ADO.NET é constituída de um conjunto de bibliotecas orientadas a objetos da plataforma .NET, que permite a interação e o acesso a fontes de dados. Normalmente, uma fonte de dados é um banco de dados, mas poderia ser também um arquivo texto, uma planilha Excel ou um arquivo XML. No contexto desse trabalho, vamos considerar o ADO.NET como uma forma de interação com um banco de dados, que permite realizar

tarefas relacionadas com o acesso e a manutenção de dados através de suas classes. As classes da tecnologia ADO.NET estão armazenadas no programa (*namespace) System.Data*. A denominação ADO é proveniente do texto em inglês "***A**ctiveX **D**ata **O**bjects*", que significa objetos de dados *ActiveX*. A denominação .NET indica que a tecnologia pertence a plataforma .NET.

ADO.NET é feito de um conjunto de classes, que são usadas para conectar com um banco de dados, disponibilizando acesso a dados relacionais, XML, dados de aplicações e ainda permite recobrar resultados. A tecnologia ADO.NET permite uma conexão fácil com qualquer fonte de dados com destaque para os gerenciadores de banco de dados relacionais como o SQL Server, o Oracle, MySQL entre outros.

Os componentes ADO.NET foram projetados para tratar o acesso e a manipulação dos dados. ADO.NET inclue vários objetos, que podem ser usados para se trabalhar com dados. Os seguintes componentes dão uma idéia do que pode ser feito com dados quando se usa o ADO.NET.

Segundo Microsoft (2009), os principais componentes do ADO.NET são DataSet e o provedor .NET que é um conjunto de componentes que inclui os objetos Connection, Command, DataReader, and DataAdapter. A arquitetura ADO.NET pode ser visualizada na Figura 2-7.

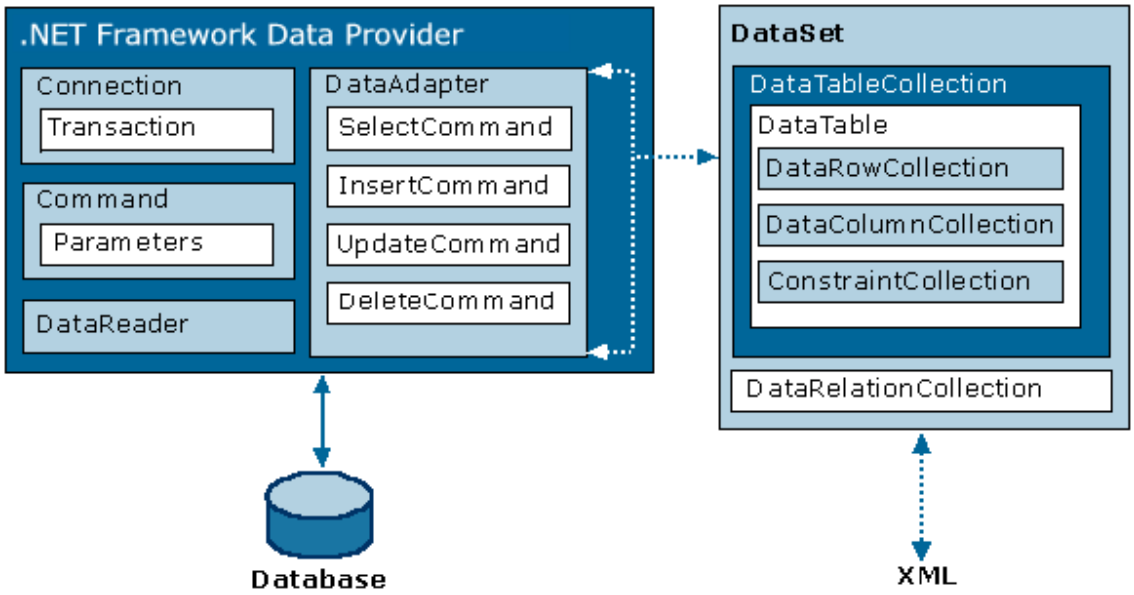


Figura 2-7 - Arquitetura ADO.NET. Extraída de Microsoft (2009).

2.4.5.1 Classe *DataSet*

A classe *DataSet* são representações dos dados em memória. Ela contem objetos múltiplos do tipo *"Datatable",* que são compostos por linhas e colunas de dados, assim como chaves primárias, extrangeiras, restrições e informações sobre relações dos dados armazenados. Os objetos da classe *DataSet* foram especialmente projetados para ajudar no gerenciamento de dados em memória e para dar suporte à operações desconectadas nos dados quando essa situação aparece. Pode-se classifica-lo como o principal componente da arquitetura desconectada do ADO.NET. Segundo Macoratti (2009), o objeto DataSet é baseado em XML e fornece as principais funcionalidades para criar aplicações para banco de dados desconectados. Permite o acesso, exibição e manipulação dos dados. É independente da fonte de dados e representa uma cópia do banco de dados em memória. O *DataSet* é um objeto usado por todos Provedores de Dados(*Data Providers*), e é por isso que ele não tem um prefixo específico de um Provedor de Dados.

2.4.5.1.1 Objeto *DataTable*

Um objeto *DataTable* é um objeto, que representa uma ou mais tabelas de dados de um banco de dados em memória. Um objeto *DataTable* está contido em um objeto *DataSet* e/ou *DataView*.

2.4.5.1.2 Objeto *DataView*

O objeto *DataView* permite a ligação de uma fonte de dados com a interface do usuário. Os dados podem ser filtrados, ordenados e pesquisados. O *DataView* permite que o usuário tenha diversas visões de um mesmo conjunto de dados. O *DataView* é utilizado para mostrar uma visão dos dados contidos em um objeto *DataTable*. A Figura 2-8 destaca o objeto *DataView* criado na interface gráfica da classe `CdTcNormaParafuso` (criada neste trabalho e descrita na seção 4.3.10).

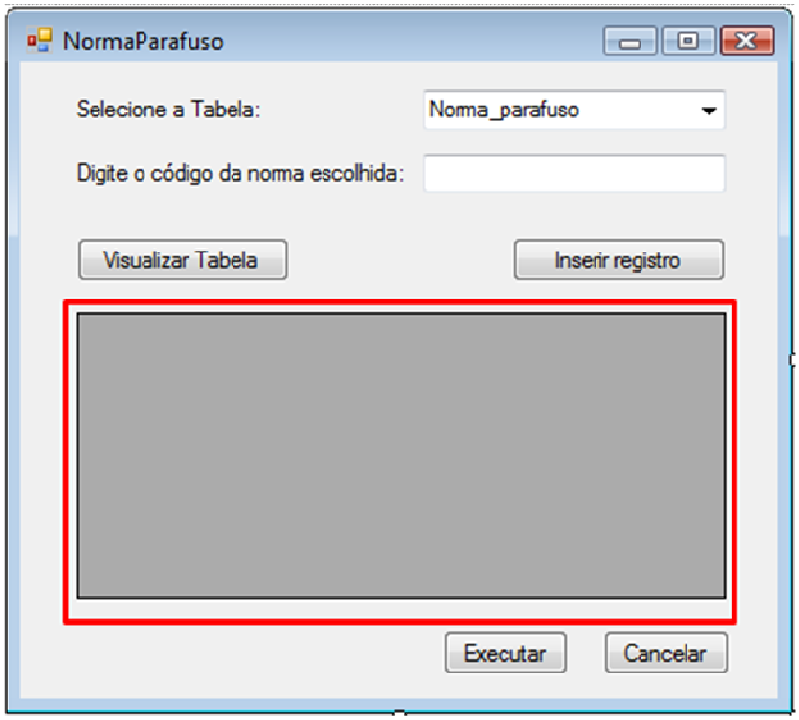


Figura 2-8 - Objeto DataView

2.4.5.2 Provedores de Dados (*Data Provider*s) *ADO.NET*

ADO .NET permite a interação com diferentes tipos fontes de dados e bancos de dados, mas não possui um conjunto único de classes, que permita essa interação de uma maneira universal. Cada fonte de dados apresenta o seu próprio protocolo de comunicação e por isso é necessário usar o protocolo específico para cada fonte de dados. *ADO .NET* disponibiliza uma forma de interação relativamente simples com fontes de dados através de conjuntos de bibliotecas para cada maneira de interagir com a fonte de dados. Essas bibliotecas são chamadas de *Data Providers,* cujo nome é usualmente relacionado com o protocolo ou com o tipo da fonte de dados, com os quais eles podem interagir. Os Provedores de Dados *ADO .NET* são bibliotecas de classe, que viabilizam a interação com protocolos de fontes de dados específicas. Os prefixos das APIs das bibliotecas indicam, que tipo de provedor eles suportam. Alguns provedores de dados (*Data Providers*) bastante conhecidos, o prefixo da API utilizado e o tipo de fonte de dados, com ao quais permitem a interação são listados a seguir:

- *ODBC Data Provider* – Prefixo API: Odbc; Tipo de Fonte de Dados: Fontes de Dados com interfaces ODBC (banco de dados antigos).
- *OLeDb Data Provider* - Prefixo API: OleDb; Tipo de Fonte de Dados: Fonte de Dados com interfave OleDb (Access, Excel).
- *Oracle Data Provider* - Prefixo API: Oracle; Tipo de Fonte de Dados: Banco de Dados Oracle.
- *SQL Data Provider* - Prefixo API:Sql; Tipo de Fonte de Dados: MS Sql Server.

Para exemplificar o uso do prefixo de uma API podemos usar objeto "*connection*", o qual será detalhado a seguir. *Connection* é um objeto que permite o estabelecimento de uma conexão com uma fonte de dados. Se for utilizada uma interface OleDb para conectar com o provedor de dados *OleDb Data Provider* o objeto *connection* a ser utilizado será o O*leDbConnection*. De maneira similar, os nomes dos objetos para a fonte de dados Odbc e para o bando de dados *SQL Server* seriam respectivamente *OdbcConnection* e *SqlConnection.*

2.4.5.2.1 Objeto Connection

Para interagir com um banco de dados é preciso ter uma conexão com ele. O objeto *Connection* é responsável pela geração da conexão do aplicativo criado com a fonte de dados. Ele ajuda na identificação do servidor do bando de dados, do nome do banco de dados, do nome do usuário, senha e outros parâmetros necessários para conectar com o banco de dados. Um objeto "*connection"* é usado por comando de objetos para saber em qual banco de dados executar o commando.

Para que seja estabelecida uma conexão com a fonte de dados, é utilizada a propriedade C*onnectionString* do objeto *Connection,* que é responsável por armazenar o texto de conexão utilizado para que a conexão seja aberta. Objetos podem ser utilizados para receber e enviar dados para a fonte de dados uma vez que a conexão esteja aberta.

No código abaixo pode ser visualizada a criação de um objeto de conexão, o preenchimento de sua sequência alfa-numérica (*string)* de conexão, sua abertura e finalmente o seu fechamento.

```
        static OleDbConnection dataConnection = new OleDbConnection();
        dataConnection.ConnectionString                                    =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
        dataConnection.Open();
        dataConnection.Close();
```

2.4.5.2.2 Objeto *Command*

Para interagir com um banco de dados é preciso especificar as ações desejadas, que é feito com um comando objeto (Command Object). O objeto *Command* é utilizado para executar declarações em SQL no banco de dados. Permite acesso a comandos do banco de dados, que retornam e modificam dados.

Um objeto "*command*" usa o objeto "*connection*" para descobrir qual banco de dados deve conectar. Um comando só pode ser criado se alguma conexão já tiver sido criada. Os

principais métodos do objeto *Command* são: *ExecuteReader, ExecuteNonQuery* e *ExecuteScalar.*

O método ExecuteReader executa declarações SQL, que retornam linhas de dados como, por exemplo, na instrução *SELECT*. O método *ExecuteNonQuery* executa declarações SQL, que não retornam dados como, por exemplo, nas instruções *INSERT, UPDATE, DELETE* e SET. Já o método *ExecuteScalar* retorna um valor único como resultado de uma função agregada como, por exemplo, nas instruções SUM, AVG, COUNT, MAX e MIN.

Os dados retornados pelos métodos do objeto *Command* só podem ser recebidos e manipulados se forem utilizados objetos *DataReader*.

O código abaixo permite a visualização da criação de um objeto *Command*, a definição de sua conexão, a definição de seu texto e finalmente a sua execução através da criação de um objeto *DataReader*.

```
OleDbCommand dataCommand = new OleDbCommand();
dataCommand.Connection = dataConnection;
dataCommand.CommandText = " SELECT * FROM Barra"
OleDbDataReader datareader = dataCommand.ExecuteReader();
```

2.4.5.2.3 Objeto *DataReader*

O objeto *DataReader* permite a leitura dos dados retornados de uma declaração *SELECT* do objeto *Command*. Várias operações com dados requerem a obtenção de uma sucessão ou fluxo de dados para leitura. Por uma questão de desempenho, os registros de dados retornados de um objeto *DataReader* são acessados e percorridos somente no modo leitura de fluxo de dados para frente. Isto implica que só se pode extrair dados da fonte de uma maneira sequencial, não sendo permitido o acesso desconectado e alterações ou atualizações da fonte de dados original. *DataReader* tem bom desempenho apenas para leitura, mas se houver necessidade de manipulação dos dados é melhor utilizar o *DataSet*.

No código abaixo, o *loop while* percorre toda a tabela torre, conforme informado no texto do objeto *Command*, linha por linha.

```
OleDbCommand dataCommand = new OleDbCommand();
dataCommand.Connection = dataConnection;
dataCommand.CommandText = "SELECT * from Torre";
OleDbDataReader datareader = dataCommand.ExecuteReader();

while (datareader.Read())
{
      // Código a ser executado, linha por linha.
}
```

2.4.5.2.4 Objeto *DataAdapter*

Quando se trabalha com dados, que são primordialmente para leitura (*read-only*) raramente é necessário modificar a fonte de dados subjacente. Algumas vezes é importante a colocação de dados em área de memória rápida (cache), para minimizar o número de chamadas ao banco de dados para acesso a dados que não mudam. O objeto *DataAdapter* torna fácil lidar com essas situações ajudando no gerenciamento e acesso de dados de modo desconectado. O *DataAdapter* preenche o objeto *DataSet* quando faz a leitura de dados e os escreve em um lote único, quando faz a persistência das mudanças de volta ao banco de dados. Uma referência para o objeto *Connection* fica contida em um objeto *DataAdapter,* que abre e fecha automaticamente a conecção quando lê do banco de dados ou escreve para ele. É o objeto que faz a ligação entre o objeto *DataSet* e a fonte de dados. Além disto, o *DataAdapter* contem referências do objeto *Command* para as operações *SELECT, INSERT, UPDATE* e *DELETE* nos dados. Em outra palavras, o *DataAdapter* utiliza objetos *Command* para executar comandos SQL na fonte de dados para preencher o objeto *DataSet* com dados e para reconhecer as mudanças feitas nos dados do *DataSet* para atualizar a fonte de dados. Um *DataAdapter* é definido para cada tabela no objeto *DataSet* e ele cuida de toda comunicação com o banco de dados, bastando indicar ao *DataAdapter* quando carregar dados do banco de dados ou escrever para o mesmo.

O código abaixo, executado quando o botão “Visualizar Tabela” na Figura 2-8 é clicado, exemplifica a utilização do objeto *DataAdapter* como ligação com o objeto *DataSet* e *DataView*.

```
            // Inicializa variáveis
          OleDbDataAdapter da;
          DataSet ds;

            // Pega a conexão que foi inicializada junto com o programa
          OleDbConnection dataConnection =
TowerCAD.Initialization.getConnection();

          ds = new DataSet();
          da = new OleDbDataAdapter("Select * from " +
comboBoxTabelas.Text, dataConnection);
          da.Fill(ds, "Tabela");
          dataGridView1.DataSource = ds;
```

Os objetos DataAdapter e DataSet são inicializados. O objeto DataAdapter recebe as instruções SQL, faz a conexão como a base de dados e posteriormente preenche o objeto DataSet. O objeto DataView recebe então o objeto DataSet. A visualização poderá ser feita então pelo usuário através da interface gráfica.

2.5 API .NET para o AutoCAD

A API (Interface para Programação de Aplicativos) .NET para o AutoCAD permite que se costumize a plataforma gráfica AutoCAD. Essa interface de programação do AutoCAD está disponível desde a versão 2004. A exposição de vários conjuntos de objetos e bibliotecas abertos para programação possibilita a manipulação do AutoCAD praticamente com as mesmas potencialidades de costumização da API ObjectARX, com exceção da derivação de classes nativas. Essa arquitetura aberta do AutoCAD pode ser acessada por várias linguagens e ambientes de programação.

O desenvolvimento de uma interface de programação (API) .NET para o AutoCAD, como a desenvolvida para essa dissertação, apresenta muitas vantagens, que serão descritas a seguir.

A API .NET para o AutoCAD foi construída com base em blocos de objetos abertos para programação. Esses objetos são reunidos em agrupamentos (*assemblies*) e contêiners (*namespaces*) representando partes do AutoCAD. Alguns exemplos de tipos de objetos na API .NET para AutoCAD são:

- Linhas, arcos e texto
- Definições/especificações de estilos
- *Layers* e blocos entre outros

Uma interface de programação desenvolvida através da API. NET para o AutoCAD interage com o AutoCAD na forma de uma biblioteca de vínculo dinâmica (DLL). Essa interação com AutoCAD pode ser feita manualmente usando o comando *NETLOAD* na linha de comando para carregar a DLL ou carregando a DLL no AutoCAD através de teclas de requisição definidas para cada aplicativo.

2.5.1 Componentes da API .NET para o AutoCAD

Segundo Autodesk (2009), a API .NET para o AutoCAD é construída com diferentes arquivos DLL que contém uma grande variedade de classes, estruturas, métodos e eventos, que permitem o acesso a objetos em um arquivo de desenho ou a um aplicativo AutoCAD. Cada arquivo DLL define diferentes contêiners (*namespaces)* que são usado para organizar os componentes da biblioteca baseado em sua funcionalidade.

Os três arquivos DLL que são freqüentemente utilizados são:

- AcDbMgd.dll. Utilizado quando se trabalha com objetos de arquivos de desenho.
- AcMgd.dll. Utilizado quando se trabalha com o aplicativo AutoCAD.
- AcCui.dll. Utilizado quando se trabalha com arquivos de costumização.

Uma referência do arquivo DLL deve ser criada no projeto para que as classes, estruturas, métodos e eventos pertencentes a esse arquivo possam ser utilizados.

2.5.2 Qualidades

A grande vantagem decorrente do uso dessa tecnologia é a combinação de facilidade e desempenho. O desempenho de um aplicativo desenvolvido em alguma linguagem da API .NET é muito próximo do desempenho de um aplicativo desenvolvido na linguagem C++, nativa do AutoCAD, utilizando a API ObjectARX.

Dentre as vantagens do uso dessa tecnologia poderíamos citar:

- Acesso ao sistema gráfico do AutoCAD por vários ambientes de programação e em várias linguagens.
- A considerável interoperabilidade com aplicações externas.
- Sistema automático de administração de memória denominado recolhedor de lixo.
- Interligação com banco de dados é significativamente mais fácil e mais rápida com C# (ADO.NET) do que com C++.
- Integração com aplicações baseadas no Windows, como o Microsoft Excel e Word, é facilitada.
- Menor custo de manutenção no longo prazo.
- A maior facilidade de criação da interface com o usuário.
- Proporciona várias capacidades que permitem não reinventar a roda.
- Maior número de desenvolvedores permite uma maior facilidade de encontrar códigos para o desenvolvimento de programas.

2.5.3 Estrutura de um Aplicativo .NET programa

Um aplicativo desenvolvido com a API .NET para AutoCAD possui uma estrutura similar a outros programas desenvolvidos com auxílio da tecnologia .NET. O aplicativo é também criado a partir de objetos com base na programação orientada a objetos e pode ser criado em um mesmo *framework* de outra aplicação criada com auxílio da tecnologia .NET.

Dentre as diferenças encontradas, podem-se destacar a necessidade de referenciar a biblioteca do AutoCAD e ainda as linhas de código que indicam quais são os métodos responsáveis pela inicialização e definição dos comandos do AutoCAD.

Um exemplo de estrutura de um arquivo principal de um aplicativo, utilizando a linguagem de programação C#, segundo Autodesk (2009a) é mostrado abaixo.

```
using System;
using System.IO;

using Autodesk.AutoCAD;
```

```
[assembly: ExtensionApplication(typeof(TowerCAD.Initialization))]
[assembly: CommandClass(typeof(TowerCAD.CdTcCommands))]

namespace TowerCAD
{
    public class Initialization : IExtensionApplication
    {
        public void Initialize() //Inicialização do programa..
        {
        }

        public void Terminate() //Término do programa...
        {
        }
    }

    public class CdTcCommands
    {
        Public CdTcCommands()
        {
        }

        [CommandMethod("CBarra")]
        static public void CBarra ()
        {
        }
    }
}
```

As primeiras linhas que são inicializadas com a palavra *using* se referem às bibliotecas que são referenciadas pelo aplicativo. A necessidade de referenciar uma determinada biblioteca decorre da necessidade de utilizar os recursos disponibilizados por uma determinada biblioteca, no aplicativo a ser desenvolvido.

A primeira linha que contém a palavra *assembly* indica ao compilador que a classe *Initialization* do contêiner *(namespace)* TowerCAD contém o código de inicialização do aplicativo. A segunda linha que contém a palavra assembly indica que a classe *CdTcComands* do *namespace* TowerCAD contém o código dos comandos a serem executados no aplicativo.

O código principal do aplicativo se encontra dentro do contêiner *(namespace)* chamado TowerCAD. O contêiner (*namespace*) TowerCAD contém uma classe de inicialização denominada *Initialization* e uma classe que contém os comandos do aplicativo e é denominada CdTcComands. CdTc é a sigla de desenvolvedor do Cadtec que é reconhecida pela Autodesk.

A classe *Initialization* possui dois métodos principais: o método *Initialize* e o método *Terminate*. Dentro do método *Initialize* se encontra o código de inicialização do aplicativo, ou seja, o que será executado quando o aplicativo for inicializado. Dentro do método *Terminate* se encontra o código de finalização do aplicativo, ou seja, o que será executado quando o aplicativo for finalizado.

A classe `CdTcComands` possui um construtor que explicita o que será executado quando uma instância da classe for criada. Essa classe possui ainda os comandos que serão executados no aplicativo. A estrutura dos comandos é exemplificada com a criação do comando `CBarra` (criado neste trabalho e descrito na seção 4.4.1).

2.5.4 Acesso ao Banco de Dados do AutoCAD

Assim como a API ObjectARX, o AutoCAD.NET tem acesso a estrutura interna de banco de dados do AutoCAD.

A estrutura interna de banco de dados do AutoCAD é responsável pelo armazenamento de todas as informações necessárias à construção de um desenho. Assim, para imprimir na tela qualquer desenho, modificar layers, utilizar blocos ou qualquer outra modificação dentro do AutoCAD, é necessário acessar a estrutura de banco de dados interno do AutoCAD.

O acesso a estrutura de banco de dados do AutoCAD é feito a partir da criação de objetos específicos das classes existentes na biblioteca do AutoCAD, que é referenciada na criação de um aplicativo AutoCAD.NET. O código utilizado para inserir uma entidade linha na tela do AutoCAD utilizando a linguagem de programação C# é mostrado abaixo.

```
        [CommandMethod("AcrescentaLinha")]
        public static void AcrescentaLinha()
        {

      // Acessa o banco de dados e documento ativo e inicia uma transação
          Document acDoc = Application.DocumentManager.MdiActiveDocument;
          Database acCurDb = acDoc.Database;
          Transaction acTrans =
acCurDb.TransactionManager.StartTransaction();

      // Abre a tabela de blocos para leitura
          BlockTableRecord acBlkTblRec;
          acBlkTblRec =
acTrans.GetObject(SymbolUtilityServices.GetBlockModelSpaceId(acCurDb),
                            OpenMode.ForWrite) as BlockTableRecord;

      // Cria uma linha que começa em at 5,5 e termina em 12,3
          Line acLine = new Line(new Point3d(5, 5, 0),
                                 new Point3d(12, 3, 0));

     // Adiciona o novo objeto a tabela de blocos e a transação
          acBlkTblRec.AppendEntity(acLine);
          acTrans.AddNewlyCreatedDBObject(acLine, true);

     // Salva o objeto no banco de dados do AutoCAD
          acTrans.Commit();
          acTrans.Dispose();
        }
```

A primeira linha do código cria um objeto *Document* chamado acDoc e atribui a ele o documento ativo do AutoCAD. O banco de dados do documento ativo é atribuído ao objeto Database chamado acCurDb criado na segunda linha.

Na terceira linha é criado um objeto *Transaction* chamado acTrans. A transação é criada no banco de dados do AutoCAD. Transações são criadas e utilizadas para permitir o agrupamento de operações em apenas uma operação. Todas as operações presentes dentro de uma transação somente são executadas se todas as operações separadamente puderem ser executadas. As transações são iniciadas e gerenciadas pelo Gerenciador de Transações.

Um objeto de escrita da tabela de blocos, BlockTableRecord, é então inicializado e denominado acBlkTblRec. Na linha seguinte lhe é atribuído o objeto para escrita da tabela de blocos do banco de dados do documento ativo.

O objeto linha é criado e denominado acLine. Na criação do objeto são informados os pontos iniciais e finais da linha criada.

O objeto linha é adicionado ao objeto de escrita da tabela de blocos e a transação. A função *Commit* do objeto de transação é então chamada e salva as modificações feitas no banco de dados e fecha os objetos abertos. Finalmente, a função *Dispose* do objeto de transação é chamada é fecha a transação.

2.5.5 Acesso ao Banco de Dados externo

Em aplicativos criados através da tecnologia .NET o acesso a um banco de dados externo é realizado, através da tecnologia ADO.NET.

2.5.5.1 Conexão

A criação e inicialização da conexão com o banco de dados do aplicativo ocorrem quando o aplicativo é carregado no AutoCAD. O fechamento da conexão ocorre quando o aplicativo é descarregado do AutoCAD. O código da classe de inicialização localizado na classe *Initialization* do aplicativo é mostrado abaixo:

```
    public class Initialization : IExtensionApplication
    {
        static OleDbConnection dataConnection = new OleDbConnection();

        static public OleDbConnection getConnection()
        {
            return dataConnection;
        }

        public void Initialize()
        {
            dataConnection.Connect                          ionString       =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
            dataConnection.Open();
```

```
        }

        public void Terminate()
        {
            dataConnection.Close();
        }
    }
```

Pode-se observar no código acima que o objeto de conexão foi criado como um atributo da classe *Initialize* e que um método para obter esse objeto foi criado e é denominado *getConnection.*

Quando o objeto de conexão é chamado dentro de um comando, deve-se utilizar a sintaxe *TowerCAD.Initialization.getConnection()* para obtê-lo.

A *string (*sequência alfa-numérica) de conexão é criada e a conexão é aberta dentro do método *Initialize*. Dentro do método *Terminate* a conexão é fechada.

2.5.5.2 A Manipulação das Tabelas

A manipulação dos dados armazenados nas tabelas é feita pelo aplicativo através de códigos que são acionados através das interfaces com o usuário criadas no AutoCAD. As quatro principais operações realizadas pelo aplicativo são inclusão, exclusão, atualização e leitura dos dados.

2.5.5.2.1 Inclusão de Dados

A operação inclusão de dados é utilizada no método incluirDados da classe do aplicativo CdTcBancoDados. Permite incluir dados nas tabelas de um banco de dados externo.

O código apresentado abaixo cria uma conexão com o texto de conexão. Em seguida, cria um comando que utiliza a conexão criada e possui um texto com as instruções SQL que deverão ser executadas. Nesse caso, inserir na tabela Torre, no campo nome_torre, o texto "Torre tipo 1". Finalmente, o comando é executado através do método *ExecuteNonQuery*.

```
            dataConnection.ConnectionString =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
            string incluiString = @"INSERT into Torre (nome_torre) values
('Torre tipo 1')";
            OleDbCommand comandoUpdate = new OleDbCommand(incluiString,
dataConnection);
            comandoUpdate.ExecuteNonQuery();
```

2.5.5.2.2 Exclusão de Dados

A operação exclusão de dados é utilizada no método excluirDados da classe do aplicativo CdTcBancoDados. Permite excluir dados nas tabelas de um banco de dados externo. É uma operação bastante similar a operação de inclusão de dados. A diferença básica entre as duas operações é a instrução SQL do comando.

No código apresentado abaixo, uma conexão com o texto de conexão é criada. O texto que contem a instrução SQL do comando é então criado. O comando é criado com o texto que contem a instrução SQL, a conexão é atribuída a ele e ele é finalmente executado.

```
            dataConnection.ConnectionString =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
            string excluiString = @"DELETE from Barra where
Barra.[no_inicial] = 1";
            OleDbCommand comandoUpdate = new OleDbCommand(excluiString);
            comandoUpdate.Connection = dataConnection;
            comandoUpdate.ExecuteNonQuery();
```

2.5.5.2.3 Atualização de Dados

A operação atualização de dados é utilizada no método atualizarDados da classe do aplicativo CdTcBancoDados. Permite atualizar dados nas tabelas de um banco de dados externo.

Muito similar as operações de inclusão e exclusão de dados, a operação de atualização também se diferencia pela instrução SQL do comando. Ao invés de conter as palavras INSERT ou DELETE, a instrução SQL da operação de atualização de dados contém a palavra *UPDATE* no começo da instrução.

2.5.5.2.4 Leitura de Dados

A operação leitura de dados é utilizada no método lerDados da classe do aplicativo CdTcBancoDados. Permite ler dados nas tabelas de um banco de dados externo. Possui uma estrutura um pouco diferente das operações anteriores. Existe a criação do objeto *dataReader* que é responsável pela leitura dos dados.

O código apresentado abaixo é um exemplo de como o aplicativo lê informações sobre quais furos pertencem a uma determinada barra que nesse caso é a barra 2.

```
            OleDbConnection dataConnection = new OleDbConnection();
            dataConnection.ConnectionString =
                "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\\BancoDados.accdb";
            dataConnection.Open();
            OleDbCommand dataCommand = new OleDbCommand();
            dataCommand.Connection = dataConnection;
            dataCommand.CommandText = " SELECT Furo.cod_furo
                                      FROM Furo f, Barra B, Aux_Barra_Furo L
                                      WHERE f.cod_furo = L.cod_furo
                                      AND B.cod_barra = L.cod_barra
                                      AND B.nome_barra = '2'"
            OleDbDataReader datareader = dataCommand.ExecuteReader();
            datareader.Close();
            dataConnection.Close();
```

A sequência do código apresentado é a seguinte: o objeto de conexão "*dataConnection"* é criado. A "*string*"(sequência alfa-numérica) de conexão "*ConnectionString*" que pertence ao objeto "*dataConnection*" é criada e inicializada de modo a fazer a ligação com o arquivo de banco de dados. O objeto "*dataConnection*" é aberto. O comando "*dataCommand*" é criado. O texto do comando é preenchido com uma *"string*" de execução criada na linguagem padrão para banco de dados SQL, que lê quais os furos pertencem a barra 2 que estão armazenados na tabela Barra. O comando é executado através do método *ExecuteReader* do objeto "*datareader*". Os objetos "*datareader*" e "*dataConnection*" são então fechados.

2.6 Banco de Dados

Um banco de dados pode ser definido como um conjunto integrado de dados relacionados a um determinado assunto ou finalidade representando algum aspecto específico do mundo real, tendo relação lógica entre os seus componentes, armazenado sob alguma forma, com objetivo de atender uma comunidade de usuários. O bando de dados permite a reorganização dos dados e a produção de informação.

Um banco de dados é usualmente mantido e acessado por meio de um *software* conhecido como Sistema Gerenciador de Banco de Dados (SGBD). Muitas vezes o termo banco de dados é usado como sinônimo de SGDB.

O SGBD incorpora as funções de definição, alteração, recuperação, armazenamento, segurança física, controle dos dados e integridade em um banco de dados. É composto por tabelas, visões e índice. Fornece ao usuário uma visão abstrata dos dados, separando a parte lógica e física do armazenamento.

Alguns Sistemas gerenciadores de banco de dados relacionais:

- Access – Indicado para aplicações pequenas e médias. É um arquivo de dados e não um servidor de dados como os demais. Limite de armazenamento pequeno.
- SQL Server 2000 – Alto desempenho, integração Windows e Framework .NET, com suporte a linguagem XML. Indicado para grandes aplicações. Integração com os serviços de múltiplas tarefas, agendamento, monitor de desempenho e log de eventos do Windows. Reduz dependência de servidor único. Utiliza a linguagem T-SQL.
- MSDE 2000 – Versão grátis do SQL Server. Indicado para aplicações de menor porte.

Normalmente um SGBD adota um modelo de dados, de forma pura, reduzida ou estendida. Um modelo de dados é uma descrição dos tipos de informação que está armazenada em um banco de dados. Tipos de modelos:

- Conceitual - Mostra entidades e seu relacionamento. Alto nível de abstração.
- Físico - Apresenta detalhes de armazenamento interno de informações.
- Lógico - Representa a estrutura de dados de um banco de dados conforme vista pelo usuário do SGBD. Dependente de Banco de Dados.

O modelo lógico pode ser representado de algumas maneiras. Dentre elas, podemos destacar:

- Hierárquica - Os dados estão na forma de árvore
- Orientada a objetos
- Rede
- Relacional - Organiza dados em tabelas com relações entre si.

Dentre os conceitos relacionados com banco de dados, que ainda podem ser citados, podemos destacar:

- A integridade referencial, que verifica a inclusão ou exclusão de dados dependentes de outros e permite a propagação ou não das ações para outras tabelas.
- A chave primária que não permite a existência de dois registros com o mesmo valor e que só pode ser colocada apenas uma vez em uma determinada tabela.

2.6.1 O Modelo Relacional de Dados

Os princípios básicos do modelo relacional aplicam-se a qualquer banco de dados baseado no modelo relacional de dados segundo Battisti (2009).

2.6.1.1 Conceitos

As tabelas são também conhecidas como entidades na linguagem relacional, elas armazenam todas as informações de um banco de dados relacional.

As colunas das tabelas armazenam os atributos. Os atributos podem ser definidos como características individuais da tabela e também são denominados campos.

As linhas das tabelas armazenam os registros. Um registro é uma linha completa de informações formada por um conjunto de campos preenchidos.

No caso da tabela `Perfil` da Figura 2-9, podem-se observar os campos “cod_perfil”, “Aba1”, “Aba2”, “Esp”, “Peso” e “Raio”. Esses campos armazenam características presentes nos elementos Perfil como, por exemplo, os valores das abas e a espessura. A tabela possui oito registros.

cod_perfil	Aba1	Aba2	Esp	Peso	Raio
1	2000	2000	30000	88	40000
2	2000	2000	40000	114	40000
3	2500	2500	30000	111	40000
4	2500	2500	40000	145	40000
5	2500	2500	50000	177	40000
6	3000	2000	30000	112	40000
7	3000	2000	40000	146	40000
8	3000	2000	50000	178	40000

Figura 2-9 - Campos e Registros

Um campo com chave primária simples é um campo que não possui dois registros com o mesmo valor. Os valores do campo da chave primária simples são únicos. Com a chave primária simples é possível garantir que dois registros não podem ser cadastrados com o mesmo código. No caso da tabela `Perfil`, pode-se garantir que cada perfil cadastrado é único pelo campo cod_perfil.

Podem existir chaves primárias compostas, quando mais de um campo compuser a chave primária. Nesse trabalho apenas um campo será uma chave primária, ou seja, existirão somente chaves primárias simples.

Em um banco de dados relacional, quando um registro aponta para o outro, que é seu dependente, é necessário criar regras para que o registro "pai" não possa ser excluído se ele tiver registros "filhos" (as suas dependências). A finalidade do uso da integridade referencial é evitar registros órfãos e manter as referências sincronizadas, de forma que não exista nenhum registro, que faça referência a outros registros, que não mais existam. Depois de imposta a integridade referencial, qualquer operação, que pode violar a integridade referencial da relação entre as tabelas é rejeitada.

2.6.1.2 Relacionamento entre tabelas

Os relacionamentos entre tabelas existem com finalidade de fazer a ligação entre as informações, que estão armazenadas em cada uma das tabelas da estrutura de banco de dados. O conceito de relacionamento entre tabelas é um conceito fundamental para o modelo relacional de dados segundo Battisti (2009).

Existem três tipos de relacionamento entre tabelas: um para um, um para vários e vários para vários.

2.6.1.2.1 Relacionamento Um para Um

Esse relacionamento ocorre quando os campos das duas tabelas conectadas são chaves primárias de suas respectivas tabelas, ou seja, quando os campos não apresentam valores repetidos.

É recomendada, quando é necessário, particionar uma tabela com muitos campos ou isolar uma parte da tabela por motivo de segurança. Essa partição auxilia a evitar a repetição

desnecessária de informações em diferentes tabelas em algumas situações, mas não é muito utilizado porque normalmente as informações podem ser armazenadas em uma mesma tabela. Não foi utilizado nenhum particionamento na criação da estrutura de banco de dados desse trabalho.

2.6.1.2.2 Relacionamento Um para Vários

Esse relacionamento ocorre quando uma das tabelas possui um campo que é chave primária e a outra tabela possui um campo que não é chave primária, ou seja, o primeiro campo não pode ser repetido e o segundo pode ser repetido várias vezes. É o tipo de relacionamento mais comum e é utilizado várias vezes na estrutura de banco de dados desse trabalho.

Um exemplo desse tipo de relacionamento na estrutura de banco de dados criada para essa dissertação, é a ligação entre a tabela `Torre` e a tabela `Tipo_Torre` mostrada na Figura 2-10. A tabela `Torre` e a tabela `Tipo_Torre` são detalhadas no capítulo 5.

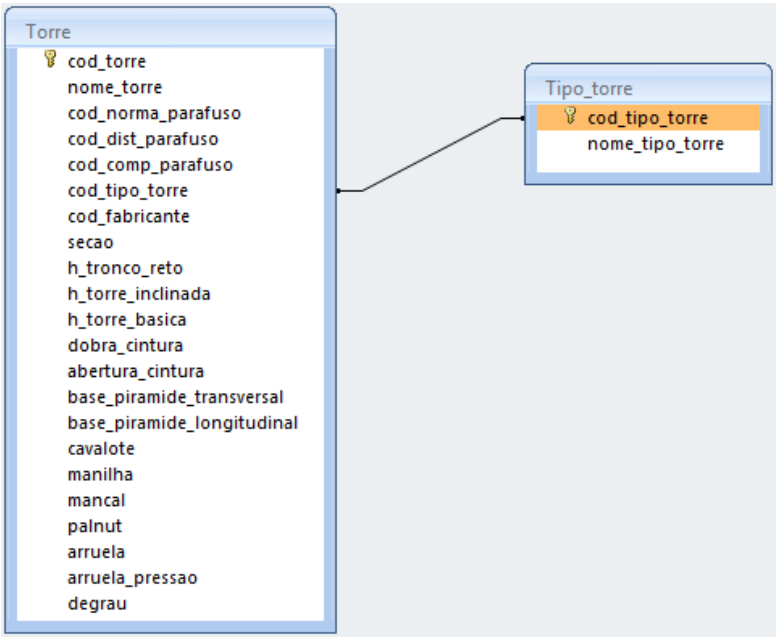


Figura 2-10 - Relacionamento Um para Vários

O campo de ligação entre as duas tabelas, denominado aqui cod_tipo_torre, não pode ter valores repetidos na tabela `Tipo_Torre`, visto que existem apenas um número finito de tipos de torres e não é permitido cadastrar dois tipos diferentes com o mesmo código. Já na tabela `Torre`, o campo cod_tipo_torre pode ser repetido várias vezes, já que várias torres podem ter o mesmo tipo e consequentemente o mesmo valor no campo cod_tipo_torre.

2.6.1.2.3 Relacionamento Vários para Vários

Um relacionamento Vários para Vários ocorre quando os valores dos campos, em ambos os lados do relacionamento, podem repetir.

Na prática, esse relacionamento geralmente não é implementado devido aos problemas que ele introduziria ao modelo de banco de dados. Caso seja necessário a implementação desse relacionamento na estrutura de banco de dados, basta criar dois relacionamentos Um para Vários com a utilização de uma tabela auxiliar.

Um exemplo de implementação desse relacionamento é o relacionamento entre a tabela que contem as informações sobre os furos e a tabela que contem as informações sobre as barras. Uma barra pode conter vários furos e um furo pode estar contido em várias barras. Foi criada uma tabela denominada `Aux_Barra_Furo` para auxiliar a criação dos dois relacionamentos Um para Vários. A tabela `Barra` é ligada à tabela auxiliar pelo campo cod_barra e a tabela `Furo` é ligada à tabela auxiliar pelo campo cod_furo. Ambos os campos são chaves primárias em suas respectivas tabelas. O relacionamento, com auxílio de uma tabela auxiliar, está mostrado na Figura 2-11.

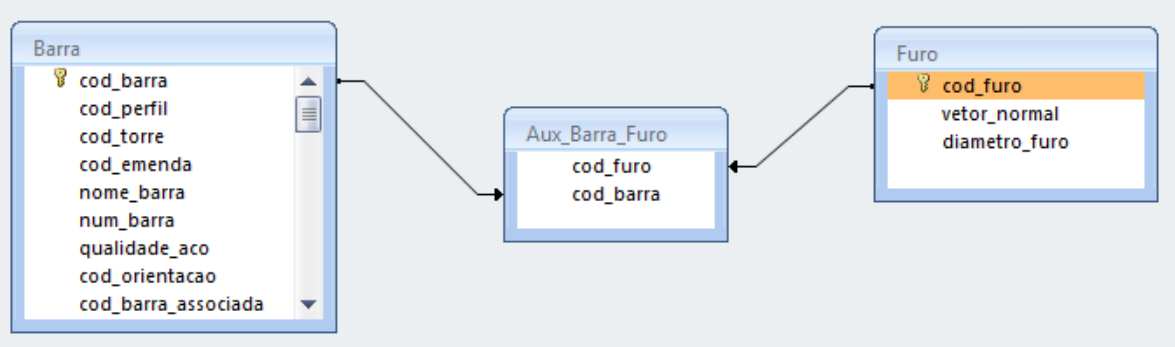


Figura 2-11 - Relacionamento tabela Furo x Barra

2.6.2 Linguagem SQL

SQL (Structured Query Language) é a linguagem padrão para manipulação de banco de dados criada pela IBM na década de 70 e padronizada pela ANSI (*American National Standarts Institute*). É uma linguagem simples para recuperar, gravar e alterar informações com segurança e rapidez.

Foi criada para permitir que usuários sem conhecimento de programação conseguissem extrair e exibir informações armazenadas em um banco de dados. Pode rodar em um simples Palmtop ou em um Mainframe.

Segundo Sacramento(2008), a linguagem SQL permite:

- Criar, Alterar e Remover todos os componentes de uma tabela;
- Inserir, Alterar e Apagar dados;
- Interrogar a Base de Dados;
- Controlar o acesso dos usuários à base de dados;
- Obter a garantia de consistência e integridade dos dados.

Além disso, a linguagem SQL possui uma série de comandos, que permitem a definição dos dados, ou seja, criação de bancos, tabelas, etc. Os exemplos desse tipo de comando são os comandos Create, Alter e Drop.

A linguagem possui ainda comandos destinados a manipulação de dados. Esses comandos permitem a criação de consultas, inserções, exclusões e alterações em um ou mais registros de uma ou mais tabelas de maneira simultânea. Os comandos Select, Insert, Update e Delete são exemplos.

No presente trabalho, os comandos para definição dos dados não foram utilizados porque a estrutura de banco de dados já estava elaborada. Foram utilizados apenas os comandos para manipulação dos dados que serão descritos a seguir.

2.6.2.1 Comando Select

O comando Select faz uma solicitação ao banco de dados e espera como retorno registros de uma ou mais tabelas. Os atributos esperados são especificados após a palavra SELECT. Caso todos os atributos sejam necessários, é utilizado o operador *. As tabelas envolvidas são especificadas após a palavra FROM. Um exemplo de um comando que pede ao banco de dados que retorne todos os atributos da tabela barra é:

```
"SELECT * from Barra"
```

2.6.2.2 Comando Update

O comando Update é utilizado para modificar (atualizar) os valores de atributos de um ou mais registros. É necessário cuidados para que sejam modificados apenas os registros desejados. No exemplo a seguir, o valor do campo cod_tipo_torre da tabela Torre é modificado para 2:

```
"UPDATE Torre SET Torre.[cod_tipo_torre] = 2"
```

2.6.2.3 Comando Delete

O comando Delete remove um registro ou um conjunto de registros contidos em uma tabela. Assim como no comando Update, atenção é necessária para campos errados não sejam modificados. O exemplo mostra a exclusão do registro da tabela barra onde o nó inicial da barra é igual a 1:

```
"delete from Barra where Barra.[no_inicial] = 1"
```

2.6.2.4 Comando Insert

O comando Insert tem como objetivo a criação de uma ou mais linhas e a respectiva inserção em uma tabela. O número de valores, a serem inseridos, especificados na instrução do comando não pode ser maior do que a quantidade de campos existentes na tabela. A instrução a seguir insere na tabela Torre, no campo nome_torre o valor Torre tipo 1:

```
"insert into Torre (nome_torre) values ('Torre tipo 1')"
```

3. O PROJETO DE TORRES

3.1 CADTEC – Centro Avançado de Desenvolvimento Tecnológico e Ensino da Computação gráfica

O CADTEC é um Centro de Desenvolvimento Tecnológico pertencente ao Departamento de Engenharia de Estruturas da UFMG, que desenvolve projetos de pesquisa nas áreas de computação gráfica e está equipado com os recursos necessários para esse fim. O Grupo de Pesquisa CADTEC está certificado pelo CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), e o CADTEC é também membro da rede mundial de desenvolvedores de aplicativos da Autodesk (ADN – Autodesk Developers Network's).

Os trabalhos desenvolvidos no CADTEC são focados em projetos de pesquisa na área de computação gráfica com destaque para a automação de processos via tecnologia CAD/CAE/CAM/CIM. As pesquisas desenvolvidas no CADTEC tiveram como produtos dissertações e tese e sistemas computacionais (software) para automação de processos, que despertaram o interesse da comunidade científica e das indústrias por esta linha de pesquisa.

Vários trabalhos foram realizados em parceria com empresas. As parcerias têm como objetivo melhorar a produtividade da indústria e exercer o papel da comunidade acadêmica como propulsora do desenvolvimento de tecnologias no setor produtivo. Dentre as empresas que já fizeram parcerias com o CADTEC, podemos destacar CODEME, PREMO e CR Gontijo.

3.2 O Projeto de Torres Metálicas

3.2.1 O Processo Produtivo

O processo produtivo de torres metálicas é dividido em diversas etapas. O fluxograma da Figura 3-1 permite visualizar todas as etapas e sua ordem na seqüência do processo, que vai da definição das especificações até a fabricação das estruturas.

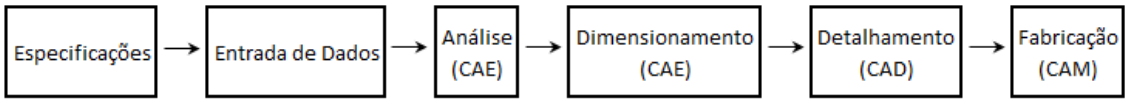


Figura 3-1 - Fluxograma do Processo

Estas etapas podem ser automatizadas por sistemas computacionais CAD, CAE e CAM. As tecnologias CAD normalmente são aplicadas na modelagem e no detalhamento das estruturas, as tecnologias CAM são aplicadas na automação da etapa de fabricação e produção de produtos e as tecnologias CAE são aplicadas nas etapas de análise e de dimensionamento.

Uma descrição de cada etapa do processo é feita nos sub-tópicos a seguir.

3.2.1.1 Concepção e Especificações

A concepção de uma torre começa pela etapa de especificações, que é a primeira etapa do processo produtivo de torres metálicas. As especificações são definidas pelo cliente do projeto ou por uma empresa contratada com base em suas preferências de projeto, disponibilidade de mercado e outras variáveis considerações. O calculista e o projetista da torre devem seguir as especificações definidas pelo cliente durante toda a execução do projeto.

A definição de quais tipos de estruturas serão fabricadas e quais os carregamentos serão suportados pelas torres é feita pelo cliente do projeto com base no perfil topográfico da linha, na voltagem que deverá ser transportada, nas distâncias elétricas e outras variáveis. O calculista da torre recebe o projeto básico da estrutura com dimensões básicas e as cargas que deverão ser suportadas conforme exemplificado pela Figura 3-2. As indicações de

quais serão as normas que deverão ser utilizadas para calcular a torre também são definidas pelo cliente.

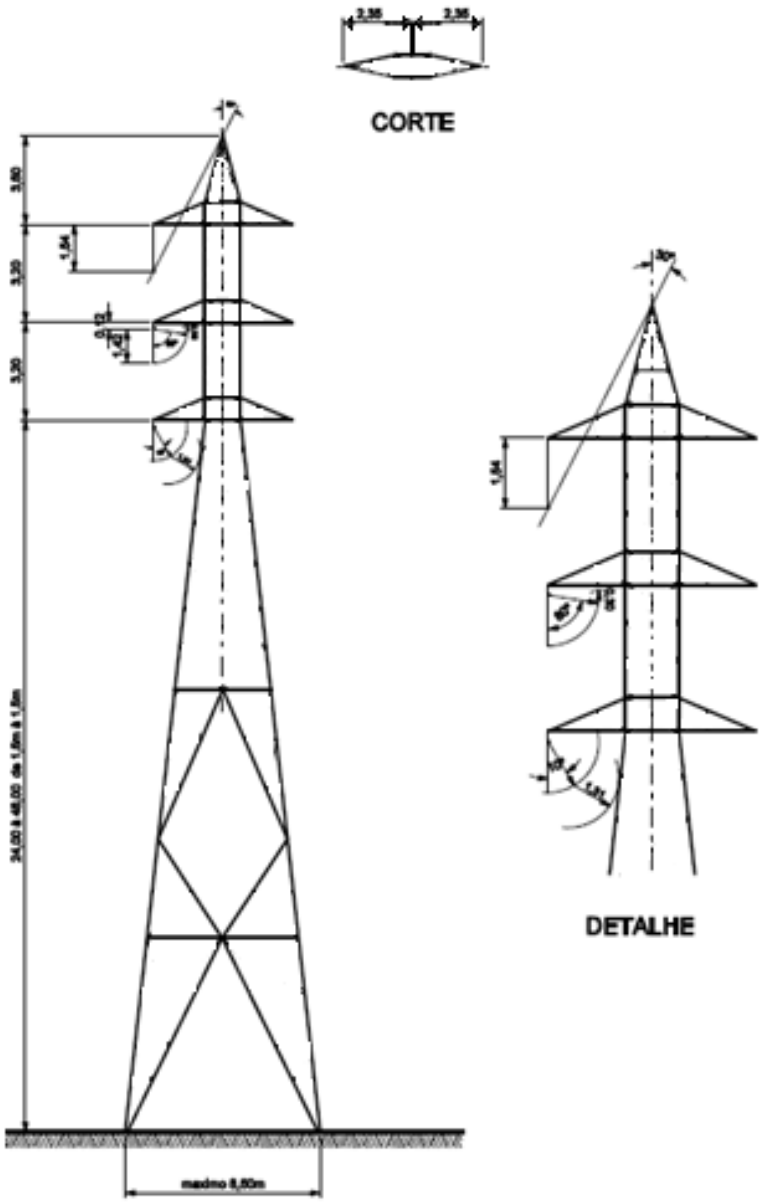


Figura 3-2 - Projeto básico da estrutura

Ao projetista são repassadas as informações sobre tipos de detalhes, que devem ser utilizados, materiais que deverão ser utilizados para fixação dos cabos, distâncias básicas de projeto, galvanização, comprimento máximo de um perfil, placas de sinalização e outras informações.

3.2.1.2 Modelagem Geométrica e Computacional

A etapa de modelagem geométrica, seja ela gráfica ou simplemente uma modelagem numérica, constitue da geração de dados geométricos e do material para serem utilizados nas etapas subseqüentes. Esses dados são gerados para a representação gráfica e para modelagem computacional como dados de entrada para a etapa do cálculo ou análise estrutural e lançamento da estrutura. Nós, barras, carregamentos, restrições e as propriedades dos elementos são os dados básicos informados.

A entrada de dados da estrutura pode ser feita via arquivo texto, banco de dados ou ainda via alguma plataforma gráfica. A entrada gráfica de dados via sistema CAD tem a grande vantagem de ser mais intuitiva e amigável com o usuário.

3.2.1.3 Análise Estrutural

A etapa de análise estrutural é a etapa na qual os esforços e deslocamentos nas barras são calculados. No caso das torres, como são treliças espaciais, existe apenas esforços de tração e compressão nas barras.

Usualmente um modelo unifilar é utilizado para fazer a representação de uma estrutura através de um conjunto de nós e barras representadas pelos seus eixos. Os nós representam as ligações entre elementos lineares e as barras representam os elementos lineares, geralmente cantoneiras no caso de torres. Os elementos são submetidos às condições de contorno. Os nós recebem os carregamentos e as liberações existentes. Em estruturas treliçadas de torres os nós recebem carregamentos e restrições. São consideradas ainda as propriedades físicas e geométricas dos elementos.

O cálculo é feito através de programas de computador pela tecnologia CAE ou engenharia auxiliada por computador.

3.2.1.4 Dimensionamento

A etapa de dimensionamento utiliza os resultados obtidos na etapa de análise estrutural para dimensionar as barras e ligações de estrutura novamente pela tecnologia CAE. O cálculo é também feito através de programas de computador de acordo com as normas adotadas e os resultados obtidos são analisados pelo calculista. Alguns ajustes podem ser feitos.

3.2.1.5 Detalhamento

O detalhamento é a etapa do processo produtivo de torres metálicas na qual os projetistas elaboram desenhos da estrutura para fabricação e montagem a partir dos dados produzidos nas etapas anteriores. A utilização da tecnologia CAD nesta etapa é muito importante para a melhoria da produtividade. A automação desta etapa via tecnologia CAD representa um dos maiores avanços tecnológicos para qualquer processo industrial. No caso dos processos de fabricação de estruturas esta etapa constitue um grande gargalo, cuja automação representa uma melhoria importante de produtividade e competitividade do setor. Os avanços apresentados neste trabalho de dissertação para automação da etapa de detalhamento de

ligações e de componentes de torres representam sua maior contribuição científica e tecnológica pelos métodos e técnicas desenvolvidos.

A etapa de detalhamento ligações de torres recebe dados de etapas anteriores de análise e dimensionamento, relacionados aos nós da estrutura, que correspondem a um ponto único de interseção dos eixos das barras. Na prática, no entanto, a interseção de barras requer o detalhamento de uma ligação com o reposicionamento das barras para que elas possam ser montadas. Informações fornecidas pelos projetistas auxiliam o reposicionamento das barras na ligação e a geração do desenho de detalhamento com as devidas correções.

Os desenhos de detalhamento das ligações incluem atributos específicos como recortes, posições de furos, qualidade do aço utilizado, chapas e outros detalhes. O papel do detalhamento da conexão (ligação) de uma peça na outra é o de especificar e mostrar detalhes da ligação e detalhes das peças individualmente para fabricação e montagem.

Alguns cuidados devem ser tomados na etapa de detalhamento de uma estrutura. Para a montagem, por exemplo, o posicionamento das barras deve facilitar a drenagem d'água, manutenção e pintura. Devem existir dispositivos ou facilidades que permitam associações de novos perfis com objetivo de substituir os perfis ou ainda reforçar a estrutura. O diâmetro mínimo dos parafusos é 12,0 mm.

O desenho de detalhamento de uma pé de três metros de uma estrutura é mostrado na Figura 3-3.

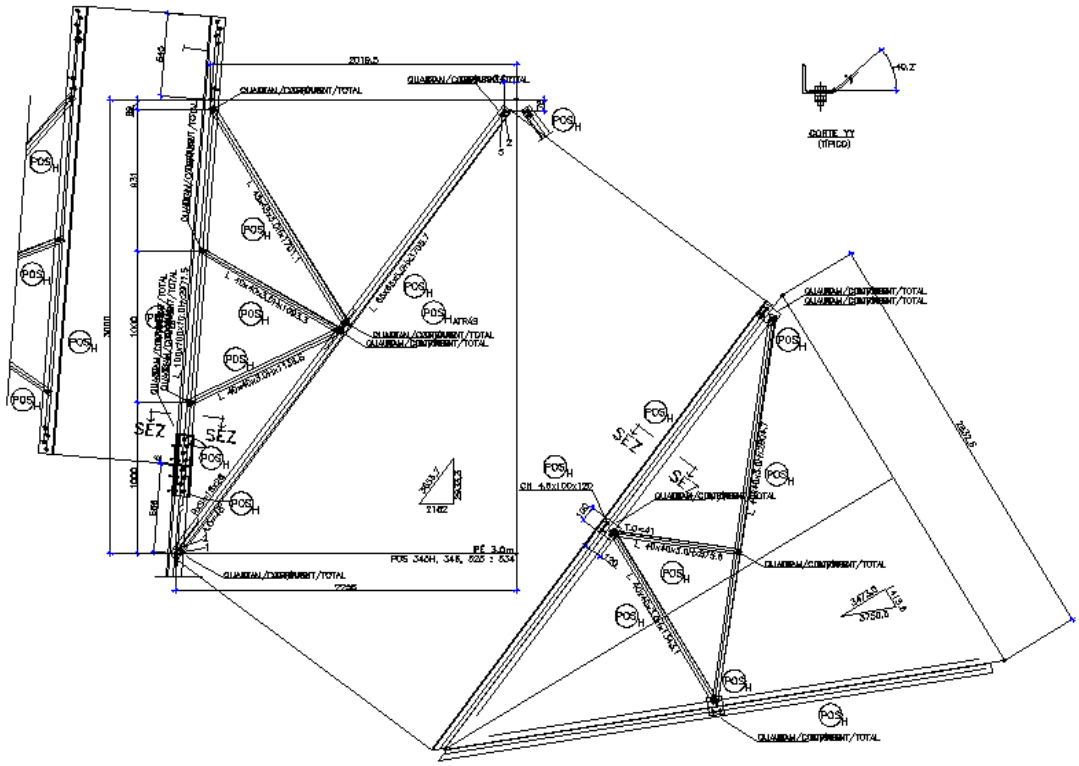


Figura 3-3 - Detalhamento de um pé de uma torre

Não existe um único padrão de detalhamento de estruturas treliçadas de aço para linhas de transmissão. Alguma padronização pode ocorrer adivinda do resultado de conhecimentos e experiências de projetistas acumulados ao longo do tempo e associado a um certo consenso de um grupo de profissionais.

Os padrões não existem para dar liberdade aos projetistas e fabricantes de levarem em conta as condições dos mercados locais e suas particularidades. O tamanho máximo das peças, distância entre parafusos e a espessura mínima são exemplos de particularidades de mercados. Em outros casos, os padrões não são definidos por falta de consenso dentro da indústria.

Silva, J.B.G.F & all (2009) compara alguns dos padrões utilizados pelos projetistas ao redor do mundo e os padrões utilizados pela indústria. Dentre os documentos analisados destacam-se:

- ASCE 10-97.
- CENELEC Standards EN 50341.
- ECCS Convenção Européia para Construções em Aço.
- Práticas e padrões brasileiros.
- Práticas e padrões italianos.
- Práticas e padrões coreanos.

O presente trabalho toma como referência as práticas e padrões brasileiros, mas possui uma estrutura adaptável a uma possível expansão para outros padrões.

3.2.1.6 Fabricação

É a última etapa do processo produtivo de torres metálicas. Consiste em fabricar todas as peças que foram projetadas para posterior montagem em campo. A Figura 3-4 mostra um conjunto de perfis fabricados e prontos para serem enviados para a montagem.

Figura 3-4 - Perfis prontos para serem montados. Extraída de Silva, J.B.G.F & all (2009)

Em geral, são utilizadas máquinas CNC para fabricação da estrutura. Essas máquinas recebem arquivos eletrônicos para que as máquinas trabalhem sem a interferência humana. Denominada tecnologia CAM ou manufatura auxiliada por computador, essa tecnologia permite que a máquina corte, fure e dobre os perfis a partir de dados numéricos. Existe uma equipe na fábrica que, a partir dos desenhos de detalhamento, fazem desenhos de croqui de cada peça, que é transformado em um arquivo eletrônico para a máquina CNC executar a fabricação da peça.

3.2.2 Integração entre as etapas do processo

A missão do sistema desenvolvido para essa dissertação é de integração das etapas de desenvolvimento de projetos de estruturas de torres metálicas através do controle e do gerenciamento de dados produzidos durante o processo, para garantir uma importação/exportação precisa e confiável. A integração entre as fases de dimensionamento e o detalhamento precisa ser gerenciada por um sistema que garanta a integridade e a consistência dos dados a serem transferidos automaticamente. O sistema deve gerenciar

também a automação da integração entre interfaces das fases de detalhamento e fabricação para permitir uma alimentação precisa das máquinas digitais de manufatura automatizada.

Segundo Leite (2007), a automação de projetos de estruturas deve ser buscada não apenas através do desenvolvimento de sistemas individualizados via tecnologias CAD, CAE e CAM, mas pelo desenvolvimento de sistemas integrados, que permitam o gerenciamento e a integridade de dados ao longo do processo de intercâmbio de informações entre as várias etapas. Esses sistemas devem fornecer interfaces compatíveis entre os sistemas CAD e os sistemas CAE utilizados no projeto, podendo ainda prever uma interface para os sistemas CAM do processo de fabricação.

Segundo Souza (1995), o resultado final de um processo, no qual o desenvolvimento de vários projetos é realizado de forma separada, não representa a solução mais adequada em termos de grau de complexidade, continuidade e de desempenho. Se um projeto não realimenta os demais ao longo do processo as dificuldades podem se propagar.

A integração de etapas, além de reduzir erros, diminui o tempo gasto na transferência de dados. As conseqüentes diminuições de erros e de desperdícios de tempo e de recursos humanos, obtidos pelo uso destes sistemas, vão refletir em ganhos financeiros e na competitividade da indústria. Aumenta assim a eficiência, melhora a produtividade e a qualidade do produto final.

Os sistemas de integração devem ser consistentes, completos e corretos, para garantir um alto grau de confiabilidade ao processo, a qualidade do produto e a eficiência das soluções adotadas.

No projeto de estruturas de torres metálicas, a falta de vinculação entre as etapas de projeto é um fator de origem de erros, atrasos e prejuízos. Apesar da existência de sistemas CAE e CAM específicos para cada uma das etapas, estes sistemas operam de forma desvinculada, sem integração com os demais sistemas do processo. Apesar dos sistemas contribuírem para diminuição de erros isoladamente, a transferência de dados entre os sistemas é geralmente um grande gerador de erros.

A falta de integração entre as atividades de um processo tem como conseqüência o comprometimento da qualidade do produto. O fator humano inserido nas atividades de transposição de dados de um sistema para o outro, que inclui manipulação, formatações e conversões de dados, aumenta a possibilidade de erros. Existe ainda o problema da propagação de erros que ocorre quando as etapas vão sendo alimentadas com dados alterados. O sistema desenvolvido para essa dissertação constitue uma importante contribuição para a automação da etapa mais crítica do processo de fabricação de torres, que é o detalhamento.

3.3 Etapas de Desenvolvimento

Para que o projeto de pesquisa pudesse ser realizado com sucesso e no tempo que foi programado, foram elaboradas etapas de desenvolvimento do projeto. A divisão em etapas permite uma maior organização e diminui as chances de falha.

O desenvolvimento do projeto de pesquisa foi dividido em quatro etapas principais: levantamento bibliográfico, qualificação computacional, qualificação técnica e desenvolvimento do aplicativo.

3.3.1 Levantamento Bibliográfico

Durante a etapa de levantamento bibliográfico foram estudados em detalhes os trabalhos já realizados na área de programação relacionados com a tecnologia CAD aplicada à engenharia estrutural, com destaque aos projetos desenvolvidos no CADTEC. Os aplicativos desenvolvidos foram estudados desde sua concepção até a sua funcionalidade final.

As novidades tecnológicas surgidas após o desenvolvimento desses trabalhos também foram levantadas e consideradas. Vários cursos de aperfeiçoamento foram feitos. Anteriormente ao trabalho, o conhecimento do autor sobre essas tecnologias era pouco.

As normas técnicas relacionadas à área foram revisadas. Dissertações e trabalhos técnicos referentes a torres metálicas foram revisados e considerados.

3.3.2 Qualificação Computacional

Devido ao fato de um aplicativo relacionado à tecnologia CAD ser específico e por conseqüência necessitar de uma qualificação específica, foi necessário passar por um programa de capacitação e de treinamento para adquirir a qualificação necessária para desenvolver o aplicativo.

Durante essa etapa, foram estudadas todas as tecnologias necessárias para o desenvolvimento do aplicativo. A linguagem C++, a API Object ARX e a biblioteca de classes MFC foram estudadas para que os projetos antigos pudessem ser estudados e ainda para fazer a avaliação de sua utilização no projeto a ser desenvolvido.

Novas tecnologias, ainda não utillizadas em aplicativos relacionados à tecnologia CAD, também foram estudadas. Dentre elas, podem-se destacar a API .NET, a linguagem de programação C# e as tecnologias relacionadas à banco de dados. O ambiente de programação Microsoft Visual Studio foi também estudado e nele foi desenvolvido o aplicativo.

3.3.3 Qualificação Técnica

Para a realização deste projeto de pesquisa foi necessário um estudo maior sobre projetos de torres de linha de transmissão, em especial, sobre a etapa de detalhamento do processo.

Essa etapa do desenvolvimento do projeto propiciou uma visão prática ao aplicativo criado. Durante alguns meses, todo o processo de produção de um desenho de detalhamento de uma torre foi acompanhado. O desenvolvimento de projetos de várias torres foram acompanhados e a execução de todo o processo como é realizado hoje foi observada com especial atenção para a etapa de detalhamento.

Durante a etapa de detalhamento foi possível identificar gargalos existentes, o que permitiu criar um aplicativo capaz de automatizar tarefas executadas pelo projetista mais suceptíveis a erros. Conseqüentemente tornando o aplicativo mais robusto e confiável e ainda capaz de aumentar a produtividade do processo.

3.3.4 Desenvolvimento do Sistema CAD

Pode ser considerada a principal etapa do projeto de pesquisa. Durante essa etapa, todo o conhecimento adquirido nas etapas de levantamento bibliográfico, qualificação computacional e qualificação técnica foi utilizado no desenvolvimento do sistema TowerCAD. O sistema foi ainda testado e aperfeiçoado com base na utilização e experiência dos projetistas. O resultado final dessa etapa é descrito a seguir.

3.4 Desenvolvimento do Sistema TowerCAD

3.4.1 Desenvolvimentos Anteriores

Os desenvolvimentos tecnológicos na área de aplicativos CAD para engenharia de estruturas, desenvolvidos no CADTEC do Departamento de Engenharia de Estruturas da UFMG, foram de grande valia para concepção desse trabalho. A concepção e as estruturas dos sistemas criadas para resolver problemas semelhantes foram minuciosamente estudadas desde os primeiros trabalhos na área, a saber, Malard (1998) e Hutner (1998), até os mais recentes desenvolvidos por Leite (2007) e Silva (2008).

Os sistemas anteriores foram orientados a objetos, desenvolvidos na linguagem C++, utilizaram a API ObjectARX, que possibilita a criação de aplicativos como bibliotecas dinâmicas, e o acesso ao código nativo da plataforma gráfica AutoCAD. Esses aplicativos foram desenvolvidos utilizando o banco de dados interno do AutoCAD para montar a estrutura de dados necessária para aplicação nos problemas para os quais foram propostos. Como conseqüência a manipulação desses dados se tornaram dependentes da plataforma gráfica, pois todos os recursos de gerenciamento do banco de dados utilizados eram da

própria plataforma. Nesta a plataforma ainda desempenhava o papel de sistema anfitrião, carregando e gerenciando a aplicação durante uma seção de trabalho.

Os trabalhos de Malard (1998) e Hutner(1998) foram pioneiros no Departamento de Engenharia de Estruturas da UFMG e no Brasil a desenvolver aplicativos associados à tecnologia CAD para automatizar os processos de análise, dimensionamento, detalhamento e fabricação de estruturas. Os trabalhos tiveram como resultado o Modelador 3D, que faz o modelamento unifilar tridimensional de estruturas metálicas prediais. A tecnologia foi repassada à empresa CODEME, que posteriormente automatizou todo seu processo de fabricação de estruturas de aço prediais desde a concepção até a manufatura.

Magalhães (2002) foi o primeiro a abordar o detalhamento de torres metálicas em sua dissertação de mestrado no PROPEEs. Seu objetivo foi automatizar mais especificamente o tronco e o stub da torre. A representação do modelo estrutural foi feita através de um modelo geométrico sólido a partir da leitura de um arquivo ASCII oriundo de um sistema de análise e de dimensionamento de torres.

Leite (2007) desenvolveu um "*framework*" que permite a geração automatizada de modelos geométricos sólidos 3D de estruturas reticuladas e a integração desses modelos com outras etapas do processo de desenvolvimento de projetos desse tipo de estrutura. Uma estrutura banco de dados baseada em classes do ObjectARX foi proposta para armazenar os dados das diferentes etapas de um projeto de estruturas reticulares. Uma grande contribuição dessa tese de doutorado foi introdução do conceito de um "*framework",* que foi concebido com base em padrões de arquitetura de sistemas computacionais com o objetivo de separar a visualização dos dados e do controle do sistema.

Foi o primeiro trabalho a apresentar o conceito de independência entre os dados e a visualização. A manipulação dos dados era feita pelo "*framework"* desenvolvido e as aplicações criadas definiam a visualização dos elementos. A escolha da plataforma gráfica a ser utilizada poderia ser flexibilizada, a partir da criação de classes de visualização.

A característica dos aplicativos anteriores era de apresentar um banco de dados para armazenamento dos dados necessários para integrar diversas etapas do processo de desenvolvimento de projetos de estruturas, que era dependente da plataforma gráfica. A conseqüência é a impossibilidade de usar uma plataforma gráfica alternativa, caso necessário.

3.4.2 Visão Geral

Esse trabalho desenvolve um sistema para automação do detalhamento de torres metálicas, que permite a integração das etapas do processo de manufatura de torres metálicas utilizando os fundamentos das tecnologias CAD/CAE/CAM. O sistema é composto por um aplicativo CAD e um banco de dados, integrados por uma interface conforme ilustrado em Figura 3-5.

Figura 3-5 - Composição do Sistema TowerCAD

A integração do sistema desenvolvido com os sistemas CAD/CAE/CAM do processo de produção de torres metálicas é obtido pelo desenvolvimento de um pré-processador e um pós-processador.

O pré-processador executa a integração com os dados da etapa de análise e de dimensionamento. Essa integração é realizada a partir de um arquivo ASCII com uma configuração pré-definida. Outras configurações podem ser definidas no futuro se assim for conveniente. O usuário localiza, seleciona e carrega arquivo no computador através de uma caixa de diálogo personalizada.

O pós-processador tem recursos que estabelecem as interfaces de comunicação entre a etapa de detalhamento e as máquinas digitais utilizadas para furo, corte e dobra das peças dentro do processo de manufatura assistida por computador (CAM). Os dados armazenados no banco de dados são transcritos e utilizados na linguagem programação das máquinas digitais.

O intercâmbio de informações no processo é consistente, completo e correto para garantir a qualidade do produto e a agilidade do processo. O sistema possibilita o preenchimento automático do banco de dados com informações das etapas do processo, garantindo assim um alto grau de confiabilidade ao sistema. O pré-processador e o pós-processador permitem o gerenciamento dos dados e dos resultados obtidos no processo sem perdas de informação nas interações entre as etapas.

A arquitetura do sistema foi projetada para abstrair as informações essenciais contidas nas diversas etapas do processo produtivo de torres metálicas conforme pode ser visualizado no fluxograma da Figura 3-6. Os dados são armazenados e gerenciados a partir do banco de dados independente da plataforma gráfica para alimentar ou re-alimentar as etapas do processo produtivo. O banco de dados, construído em um sistema gerenciador de banco de dados, é definido pelas tabelas e o relacionamento entre elas. A utilização de um banco de dados é a maneira mais eficiente para armazenar e gerenciar os dados.

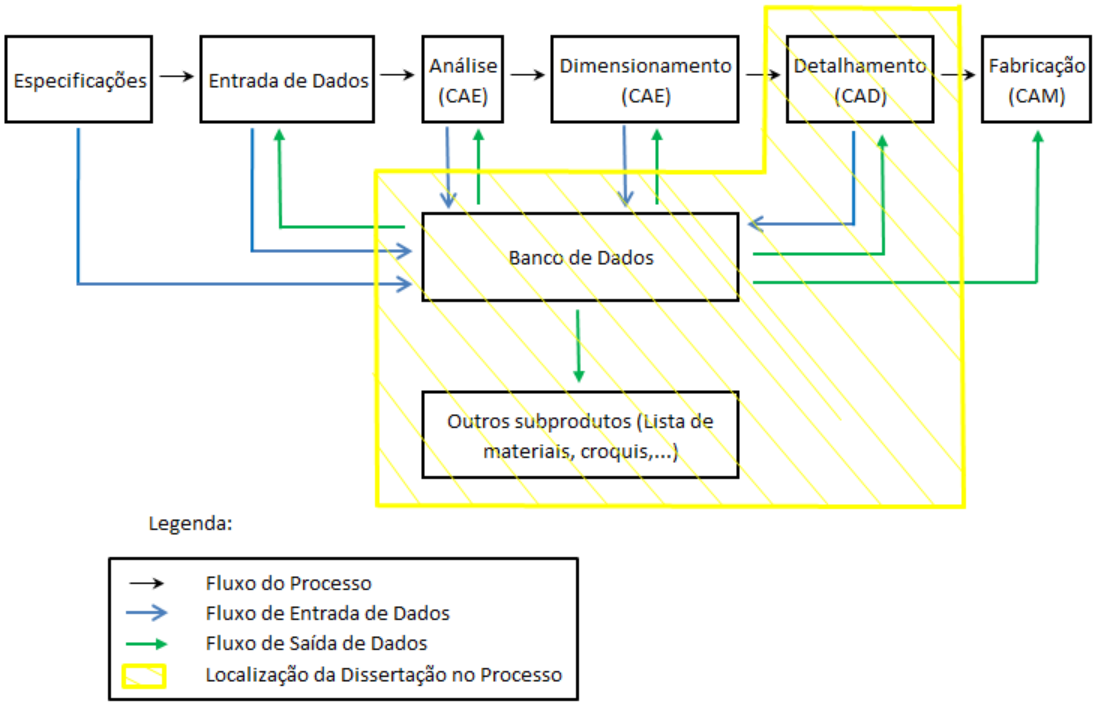


Figura 3-6 - Fluxograma do Processo

Das especificações são extraídas informações como, por exemplo, comprimento máximo do perfil, galvanização e placas de sinalização. Da entrada de dados, informações como os nós e barras. Da Análise, os esforços solicitantes. Do dimensionamento, os perfis que serão utilizados. Essas informações obtidas através de um arquivo ASCII com formato pré-definido, que é lido e compreendido pelo aplicativo.

As etapas de entrada de dados, análise e dimensionamento poderão ser re-alimentadas com as informações armazenadas no banco de dados.

Para o detalhamento de uma torre, as informações necessárias poderão ser fornecidas pelos projetistas através de interfaces gráficas amigáveis que são acessadas de dentro da plataforma gráfica e que tem conexão direta com o banco de dados externo conforme pode ser visualizado na Figura 3-7.

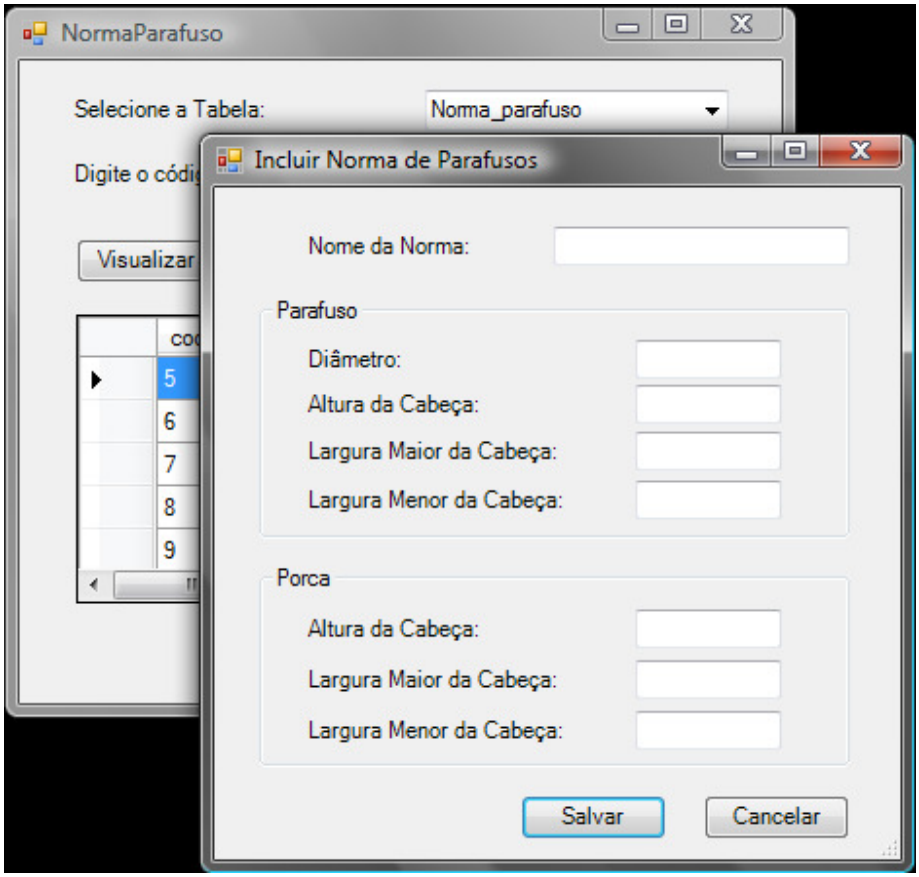


Figura 3-7 - Interfaces com o projetista

A automação do detalhamento é feita a partir de informações fornecidas e/ou armazenadas no banco de dados e na utilização da plataforma gráfica para visualização da representação gráfica.

Outros produtos podem ainda ser desenvolvidos a partir dos dados armazenados no banco de dados externo dentre os quais, podem-se destacar os croquis para fabricação**.**

3.4.3 Características

A automação de processos via sistemas CAD permite produzir desenhos de melhor qualidade com maior agilidade. O desenvolvimento do sistema TowerCAD tomou como base conceitos modernos de programação e de Engenharia Estrutural.

O sistema TowerCAD foi desenvolvido para ser flexível e independente de tal forma que interfaces simples permitam sua comunicação com sistemas CAE (*Computer Aided Engineering*) como um pré-processador e com sistemas CAM (*Computer Aided Manufacturing*) como um pós-processador do processo produtivo de torres metálicas.

O sistema é amigável e interativo, com entrada de dados via quadro de diálogos ou leitura de arquivos gerados por algum programa específico de cálculo. A utilização de um ambiente gráfico amplamente utilizado, como o AutoCAD, proporciona a criação de uma interface agradável e de fácil utilização pelo usuário.

Dentro da filosofia de desenvolvimento dos vários trabalhos já desenvolvidos no CADTEC, e na intenção de obter o melhor resultado, o sistema é facilmente atualizável para responder a futuras demandas e permitir uma posterior expansão.

Arquitetura MVC

Uma importatne contribuição desse trabalho é a utilização de conceitos e fundamentos do padrão/arquitetura MVC (Modelo-Visualização-Controle) no desenvolvimento do sistema TowerCAD. A arquitetura de 3 camadas do MVC para a interface do usuário permite separar a informação da sua representação gráfica.

Separada e independentemente o componente **Modelo** representa os dados da aplicação e as regras que regem o acesso e a manipulação desses dados. Além disso, o modelo garante a persistência das regras do negócio e fornece ao controlador a lógica do domínio para acessar as funcionalidades do aplicativo encapsuladas pelo próprio modelo.

O componente **Visualização** permite a representação do conteúdo de uma parte ou do todo do modelo e direciona as ações do usuário para o controlador. Esse componente permite também o acesso aos dados do modelo através do controlador e determina como os dados do modelo devem ser apresentados ao usuário.

O **Controlador** define o comportamento do aplicativo, interpretando as ações do usuário e as mapeando para o acesso e manipulação do modelo. As ações do usuário podem ser executadas através de botões, menus ou linhas de comando. O Controlador recebe e trata a entrada e permite a realização de ações, tais como ativação de processos ou alteração do estado do modelo e de visualização.

Em resumo, no padrão MVC o controlador escolhe o tipo de visualização a ser apresentada como resultado da solicitação do usuário, cuja ação prévia promove algum tipo de processamento do modelo.

Assim como em Leite (2007), o aplicativo é desenvolvido para ter um alto grau de independência da plataforma gráfica. Para tal, o aplicativo é desenvolvido de forma a haver uma separação entre os dados e a representação gráfica conforme indicado em Figura 3-8. Tudo o que é relacionado com dados está armazenado no banco de externo. A plataforma gráfica, nesse caso o AutoCAD, é utilizada para promover a representação gráfica do produto final.

Figura 3-8 - Representação gráfica x Dados

A independência obtida entre dados e visualização permite a flexibilização na escolha da plataforma gráfica a ser utilizada e facilita a implementação das representações gráficas, uma vez que toda a manipulação de dados é realizada pelo banco de dados externo e a aplicação apenas define a visualização dos elementos. O mesmo banco de dados pode servir de base para aplicações desenvolvidas em diferentes plataformas gráficas.

O sistema desenvolvido pode ser considerado ainda prático e funcional devido ao fato de ter sido desenvolvido em parceria com experientes projetistas de torres. O sistema busca atender ao modo de operação dos projetistas.

4. O APLICATIVO CAD PARA DETALHAMENTO DE TORRES

O sistema computacional desenvolvido neste projeto de dissertação é composto por dois componentes. Um aplicativo CAD para automação do processo de detalhamento de torres, que será apresentado a seguir, e um banco de dados construído através de um sistema gerenciador de banco de dados, acionado pelo aplicativo CAD através da tecnologia ADO.NET e que será apresentado no Capítulo 5. O Aplicativo CAD é responsável pelas classes de controle e visualização do sistema e o banco de dados pelo modelo, através do armazenamento dos dados.

Nesse capítulo é apresentada uma visão geral do Aplicativo com sua estrutura de classes, seus comandos e o template do AutoCAD no qual ele deve ser executado.

4.1 Visão Geral

Assim como os aplicativos desenvolvidos anteriormente em Hutner(1998), Malard(1998), Leite(2002), Magalhães(2002) e Leite(2007), utiliza-se, neste trabalho, a plataforma gráfica AutoCAD como anfitriã. A partir do AutoCAD, todos os recursos desenvolvidos podem ser acionados por meio de comandos que se comportam como comandos nativos.

Um dos principais aspectos que diferenciam este trabalho daqueles que o precederam é o emprego de uma nova tecnologia. A API .NET é utilizada para customizar a plataforma gráfica AutoCAD conforme Autodesk (2009). A tecnologia .NET foi escolhida por suas vantagens descritas na seção 2.5.2 aliadas à facilidade de manipulação de dados através da tecnologia ADO.NET.

A linguagem C# foi escolhida por ser originalmente orientada a objetos e concebida dentro da filosofia da tecnologia .NET conforme Sharp (2008) , aliada a este fato está a facilidade de migração a partir da linguagem C++. Este é um aspecto importante, já que C++ foi a linguagem utilizada nos projetos que precederam este trabalho. A linguagem C# é uma das principais linguagens da tecnologia .NET. Foram utilizadas ainda as tecnologias de banco de dados com auxílio da tecnologia ADO.NET e da linguagem padrão para banco de dados SQL conforme Tayor (2003) para manipular os dados e tabelas existentes no banco de dados externo. Neste ponto, encontra-se outro diferencial deste trabalho, já que em Leite (2007) foi utilizado o banco de dados interno do AutoCAD para montar a estrutura de dados do aplicativo.

4.2 Template Utilizado

Para que o aplicativo funcione corretamente, os desenhos por ele produzidos devem adotar como base um *template* do AutoCAD concebido para o aplicativo.

Template é um arquivo protótipo do que guarda configurações anteriormente definidas como *layers* e blocos. O formato do *template* é o dwt, diferentemente de arquivos de projetos produzidos no AutoCAD que tem o formato dwg. Este tipo de arquivo pode ser empregado ao iniciar um projeto para aproveitar características pré-definidades pelo usuário do AutoCAD. No caso deste trabalho foi concebido o *template* towerCAD.dwt para o aplicativo TowerCAD.

A seguir, são descritas as *layers* e blocos definidos no arquivo *template*. Quando uma destas entidades incluídas pelo aplicativo não está presente no desenho, uma mensagem irá notificar o usuário e o comando poderá não ser executado ou não será executado da maneira esperada. Isso ocorre quando o usuário utiliza o aplicativo sem inicializar o *template*. A mensagem de notificação da falta de uma layer durante a execução do comando `CBarra` (criado neste trabalho e descrito na seção 4.4.1) pode ser visualizada na Figura 4-1.

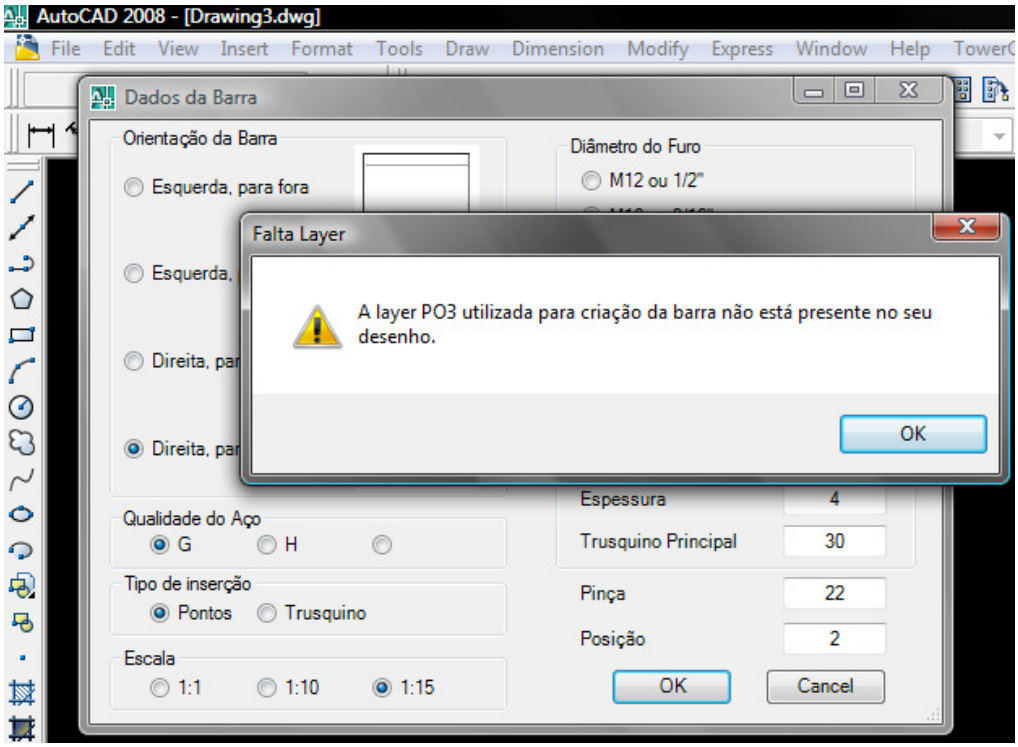


Figura 4-1 - Falta Layer

4.2.1 Layers utilizados

Em um desenho, os vários componentes são divididos em *layers* para que posteriormente seja mais fácil realizar as modificações necessárias. Para o aplicativo CAD, algumas *layers* foram definidas como padrões e deverão estar no desenho, conseqüentemente foram acrescentadas ao *template* towerCAD.dwt. As *layers* acrescentadas ao *template* foram: P01, P02, P03, P04, P05, P06, P02TR, DIM, FUROS, TXT, POSIÇÃO, LCENTRO, INDPARAF e TRUSQUINO. Todas as *layers* do template podem ser visualizadas na Figura 4-2.

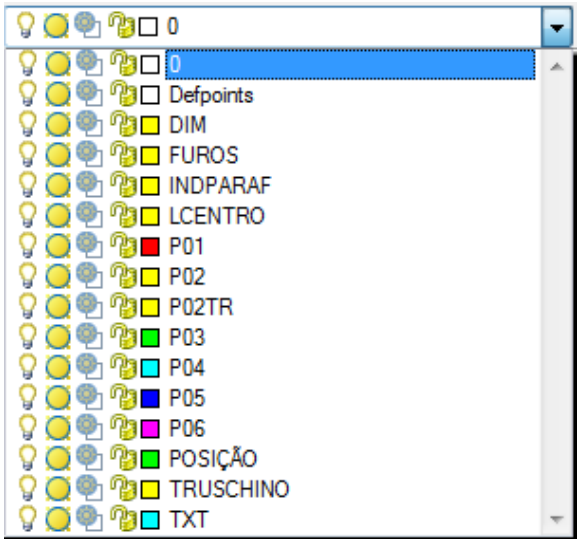


Figura 4-2 - Layers do Template

As *layers* P01, P02, P03, P04, P05 e P06 são associadas a cores. As cores associadas as *layers* são, respectivamente, vermelho, amarelo, verde, ciano, azul e magenta. Os objetos que não possuem uma *layer* definida e são desenhados em uma determinada cor, devem ser colocados em uma dessas *layers*. As cores das *layers* estão associadas a espessuras de linhas que elas estão relacionadas. A *layer* P02TR também está relacionada com cor, mas além de ser relacionada com a cor amarela, está relacionada também com o tipo de linha que é tracejado.

A *layer* DIM contém as quotas do desenho, a *layer* FUROS contém os parafusos, a *layer* TXT contém os textos e a *layer* POSIÇÃO contém todos os blocos de posição do desenho.

A *layer* TRUSQUINO é utilizada para identificação dos eixos de furação das barras, a *layer* LCENTRO é utilizada para indicação de algumas linhas de referência do desenho e a *layer* INDPARAF é utilizada para a indicação dos parafusos.

4.2.2 Blocos utilizados

Os blocos são estruturas de um desenho no AutoCAD em que podem ser agrupadas diversas entidades, às quais é atribuído um nome de identificação e um ponto para inserção. A utilização de blocos aumenta a produtividade de criação de um desenho. No *template* do aplicativo foram inseridos vários blocos com esse objetivo.

Todos os parafusos normalmente utilizados possuem blocos específicos relacionados com o diâmetro. Os blocos permitem uma identificação mais fácil e padronizada de qual parafuso está sendo utilizado. Existem blocos para os parafusos de 12 mm; 16 mm; 20 mm; 22 mm e 24 mm. Esses blocos podem ser visualizados em sequência crescente Figura 4-3.

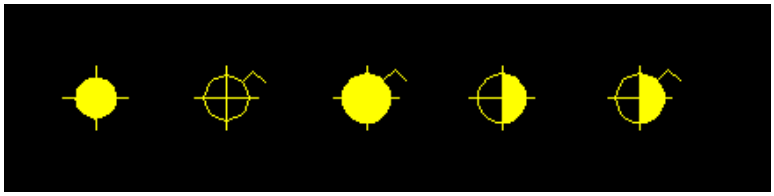


Figura 4-3 - Bloco dos parafusos

Existe ainda o bloco que representa a geometria da torre, o bloco que indica a posição da barra e o bloco que indica o parafuso degrau. O parafuso degrau é utilizado, usualmente, a partir de três metros de altura da torre e tem como finalidade permitir que uma pessoa possa subir na torre para fazer manutenção da mesma. Esses blocos podem ser visualizados na Figura 4-4.

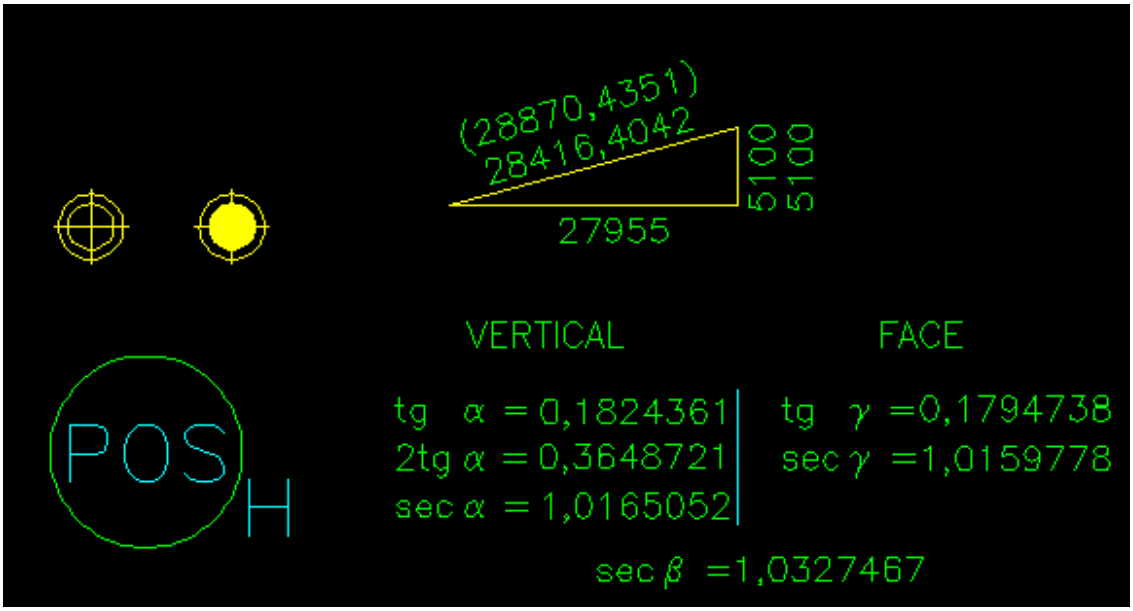


Figura 4-4 - Outros blocos

4.3 Estrutura Global do Aplicativo

O aplicativo foi dividido em vários arquivos com extensão (.cs) para que entidades sem vinculação ou relação entre si fossem separados e a busca por uma determinada função ou classe fosse mais rápida e precisa.

A Figura 4-5 mostra o *Solution Explorer* do aplicativo que contém a lista de todos os arquivos que foram criados dentro do namespace TowerCAD.

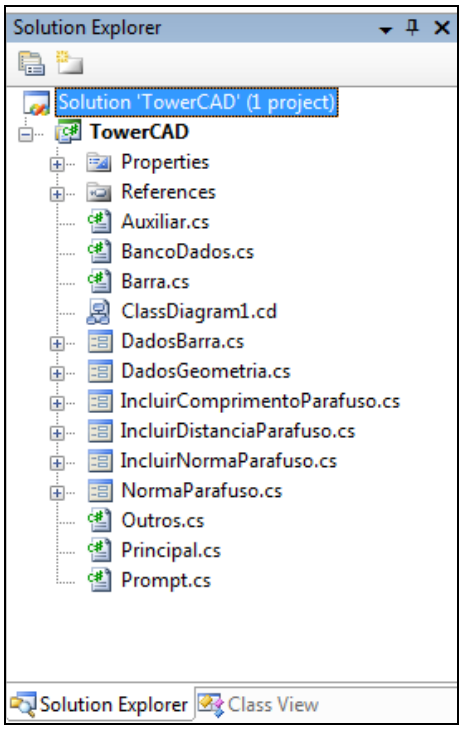


Figura 4-5 - Solution Explorer

Os arquivos `DadosBarra.cs`, `DadosGeometria.cs`, `IncluirComprimentoParafuso.cs`, `IncluirDistanciaParafuso.cs`, `IncluirNormaParafuso.cs` e `NormaParafuso.cs` armazenam classes que são derivadas da classe Windows Form. São utilizadas para fazer a interface do usuário com o aplicativo. Essas classes juntas compõem o módulo de controle do aplicativo

As outras classes também foram criadas dentro do *namespace* TowerCAD seguindo as premissas utilizadas na programação orientada a objetos, mas não são classes de interface com o usuário. Estas classes, detalhadas a seguir são: `Initialization`, `CdtcComandos`, `CdTcAuxiliar`, `CdTcBarra`, `CdTcPrompt`, `CdTcBancoDados` e `CdTcOutros`.

Todas as classes criadas no aplicativo TowerCAD podem ser visualizadas na Figura 4-6.

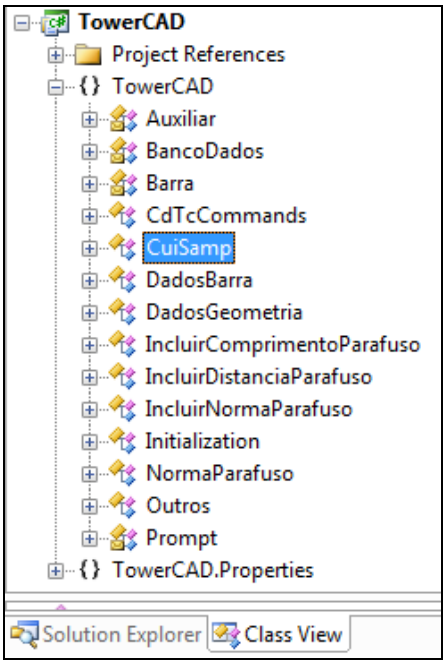


Figura 4-6 - ClassView

4.3.1 Classe `Initialization`

Localizada no arquivo `Principal.cs`, a classe `Initialization` executa o código que é acionado quando o aplicativo é inicializado ou descarregado no AutoCAD.

No aplicativo criado TowerCAD, a conexão com o Banco de Dados é aberta quando o aplicativo é inicializado e fechada quando o aplicativo é fechado. O código da classe Initialization pode ser visualizado abaixo.

```
    public class Initialization : IExtensionApplication
    {
        static OleDbConnection dataConnection = new OleDbConnection();

        static public OleDbConnection getConnection()
        {
            return dataConnection;
        }

        public void Initialize() //Inicialização do programa..
        {
            dataConnection.ConnectionString =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
            dataConnection.Open();
        }

        public void Terminate() //Término do programa...
        {
```

```
            dataConnection.Close();
        }
    }
```

4.3.2 Classe `CdTcComands`

A classe `CdTcComands` também está localizada no arquivo `Principal.cs`. As quatro primeiras letras, CdTc, referem-se ao código de desenvolvedor do CADTEC na Autodesk, empresa que comercializa o AutoCAD.

Esta classe contém todos os comandos utilizados no aplicativo. Os comandos são reconhecidos no AutoCAD ao ser digitado na linha de comandos o nome do comando ou através da utilização do menu do AutoCAD e das *toolbars*. Um comando é declarado na classe `CdTcComands` conforme se exemplifica abaixo para o caso de criação de barra (comando `CBarra`).

```
        // Define o Comando que exibe caixa de dialogo para criação de
// uma barra.
        [CommandMethod("CBarra", CommandFlags.UsePickSet)]
        static public void CBarra() // Esse método pode ter qualquer nome
        {
        }
```

A classe `CdTcComands` contém os comandos CBarra, CChapa, CCroqui, CGeometria, CPe, LerArquivo e PParafuso.

4.3.3 Classe `CdTcAuxiliar`

A classe `CdTcAuxiliar` está localizada no arquivo Auxiliar.cs. Ela contém métodos úteis que são largamente utilizados por outras classes e por comandos do aplicativo. Os métodos são, na programação orientada a objetos, como funções que estão associadas a objetos de uma classe. A Figura 4-7 permite a visualização de todos os métodos da classe `CdTcAuxiliar`.

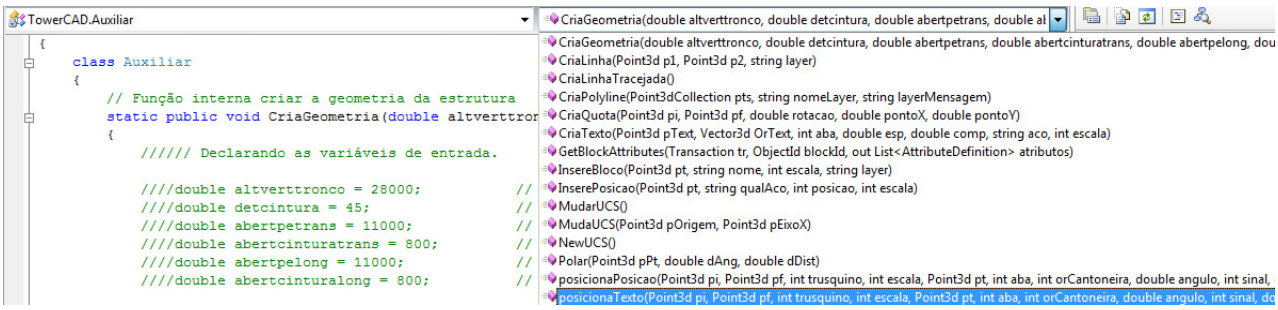


Figura 4-7 - Classe `CdTcAuxiliar`

4.3.3.1 Método `CriaGeometria`

O método `CriaGeometria` tem como finalidade o cálculo da geometria da "pirâmide" da torre. Os dados de entrada ou os parâmetros do método são: detalhe de dobra da cintura que é ilustrado na Figura 5-4 e altura vertical do tronco, abertura transversal do maior pé, abertura transversal da cintura, abertura longitudinal do maior pé e abertura longitudinal da cintura que são ilustrados na Figura Figura 4-8.

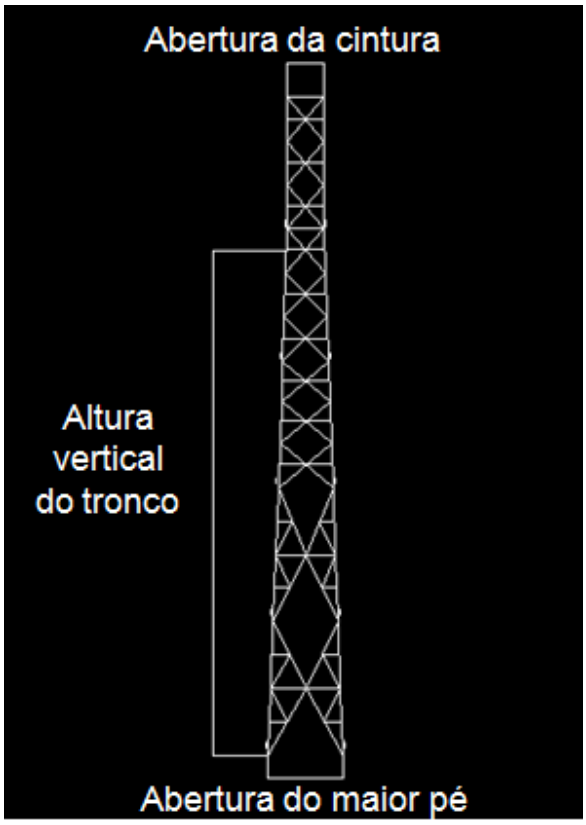


Figura 4-8 – Dados de entrada do método CriaGeometria

A altura vertical do tronco e as aberturas são informações provenientes da etapa de cálculo estrutural. A altura vertical do tronco refere-se à altura da cintura da torre até o solo com montagem mais alta da torre, ou seja, com a torre montada com a maior extensão e com o maior pé. As aberturas dos pés referem-se à distância entre os pés na torre com montagem mais alta. A indicação longitudinal e transversal existe para o caso da torre ter seção retangular. Esses dados provenientes do cálculo são armazenados na tabela `Torre` que é descrita na seção 5.3.1.

O detalhe de dobra da cintura é uma informação fornecida pelo projetista da torre e também e armazenado na tabela Torre. A Figura 5-4 permite uma melhor visualização do detalhe de dobra da cintura.

Os dados de saída do método são: Base Longitudinal, Base Transversal, Vértice, Altura na Face, tangente de α, secante de α, secante de β e secante de γ. Os ângulos e as medidas podem ser visualizados na Figura 4-9.

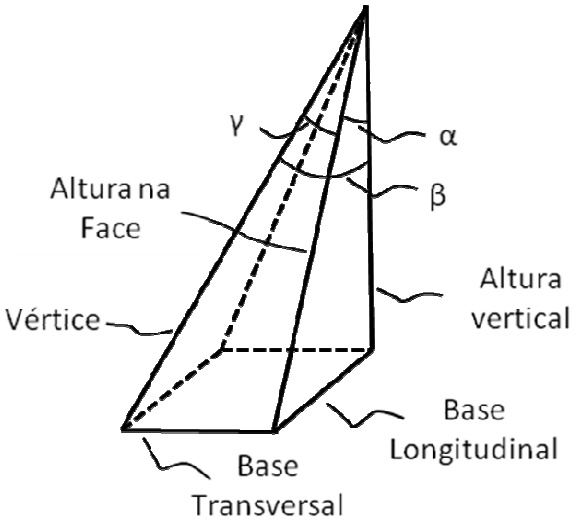


Figura 4-9 - Geometria da Torre

A partir da definição da geometria da torre é possível transformar as medidas do cálculo em medidas reais de fabricação e montagem.

4.3.3.2 Método `CriaLinha`

O método `CriaLinha` recebe como parâmetro os pontos iniciais e finais da linha além da layer que a linha será criada. Quando a *layer* escolhida não existe no desenho, uma mensagem de alerta é exibida no AutoCAD. No caso da Figura 4-10 a *layer* TRUSQUINO não estava presente no desenho, ou seja, o usuário não estava utilizando o *template* indicado.

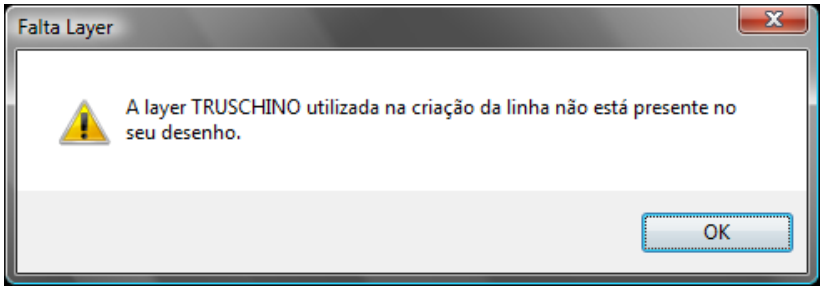


Figura 4-10 - Falta Layer

4.3.3.3 Método `CriaPolyline`

O método `CriaPolyline` cria uma polyline no AutoCAD. Uma polyline é um conjunto de linhas criadas que são agrupadas em um único objeto. O método recebe como parâmetro o conjunto de pontos que serão ligados por linhas, a *layer* que o usuário deseja que a polyline esteja e a mensagem a será exibida no caso de não existência da *layer*.

4.3.3.4 Método `CriaQuota`

O método `CriaQuota` é responsável por criar as quotas no AutoCAD. O estilo de quota adotado será o estilo que estiver sendo utilizado pelo usuário. Os parâmetros de entrada do método são os pontos iniciais e finais das quotas, o ângulo de rotação da quota e os pontos X e Y que indicam o afastamento da quota dos pontos de referência da quota.

A Figura 4-11 permite a visualização de várias quotas em vários ângulos de rotação diferentes em um desenho de um stub de uma torre. O stub é um dos tipos de fundação utilizado em torres metálicas.

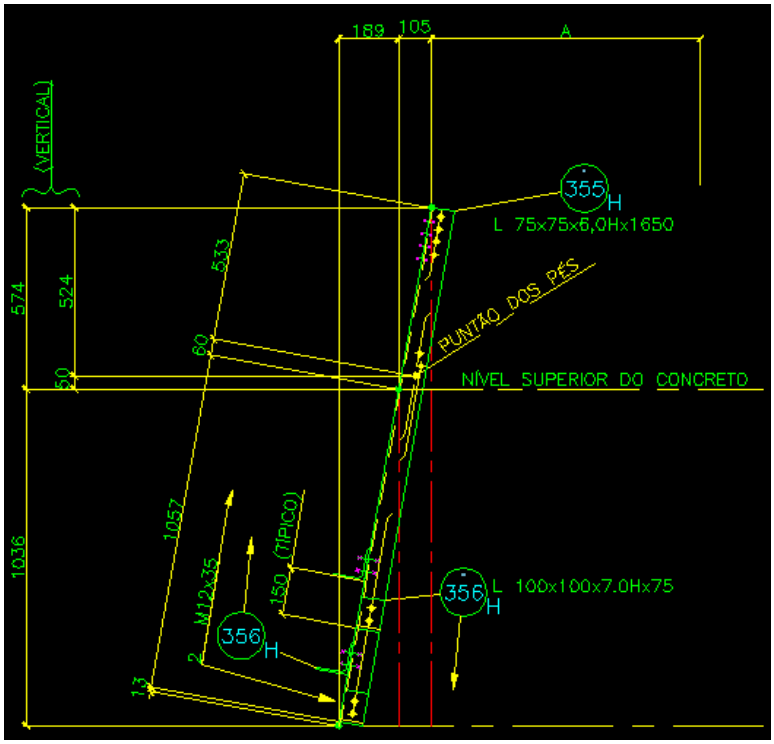


Figura 4-11 – Stub

4.3.3.5 Método `CriaTexto`

O método `CriaTexto` é responsável por criar o texto que descreve a barra. Os parâmetros de entrada desse método são o ponto de inserção do, o vetor de orientação do texto, a espessura, a aba, o comprimento, a escala do desenho e a qualidade do aço.

Foi definida a seguinte padronização para o aço:

- Quando não existe indicação o aço da cantoneira é o aço ASTM A36.
- Quando o aço é indicado pela letra H, é o aço ASTM A572 – GRAU 50.
- Quando o aço é indicado pela letra G, é o aço ASTM A572 – GRAU 60.

A visualização do texto que descreve a barra pode ser feita na Figura 4-12.

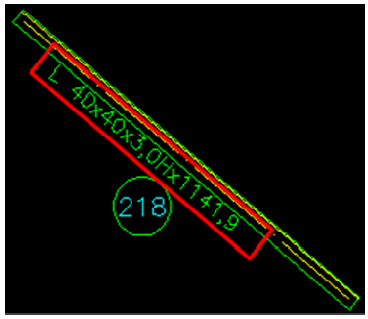


Figura 4-12 - Texto da Barra

4.3.3.6 Método `GetBlockAttributes`

O método GetBlockAttributes tem a finalidade de retornar quais os atributos são armazenados em um certo bloco do AutoCAD.

Os parâmetros de entrada do método são: a transação, o identificador do bloco e uma lista de atributos que será preenchida pelo método.

4.3.3.7 Método `InsereBloco`

O método `InsereBloco` insere na tela um bloco presente no desenho. Os parâmetros de entrada do método são: o ponto de inserção do bloco, o nome do bloco, a escala na qual o usuário deseja inserir o bloco e a *layer* escolhida. O bloco é procurado no banco de dados do desenho e caso não seja encontrado, uma mensagem é exibida na tela informando que o bloco não foi encontrado conforme a Figura 4-13.

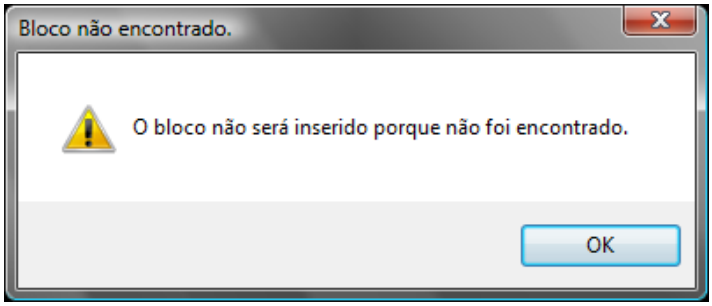


Figura 4-13 - Bloco não encontrado

4.3.3.8 Método `InserePosicao`

Apesar de `Posição` ser um bloco, foi necessário criar um método específico para esse bloco devido as suas peculiaridades. É necessário entrar com alguns outros parâmetros em relação aos outros blocos.

Os parâmetros de entrada ponto de inserção e escala são os mesmos, mas é necessário ainda incluir o número da posição e qualidade do aço. A qualidade do aço segue a mesma padronização descrita no método `CriaTexto` e o número da posição refere-se ao número de identificação da peça para montagem e fabricação.

O bloco posição pode ser visualizado na Figura 4-14.

Figura 4-14 - Bloco Posição

4.3.3.9 Método `MudaUCS`

UCS é o sistema de coordenadas do usuário da plataforma gráfica AutoCAD, em inglês, "*User Coordinate System*". Diferentemente do sistema de coordenadas globais, o sistema de coordenadas do usuário pode ser modificado pelo usuário. A modificação do sistema de coordenadas do usuário facilita o desenho, pois permite o desenho em 2D em qualquer plano do espaço de modelagem.

O método `MudaUCS` permite modificação do sistema de coordenadas do usuário de acordo com a necessidade. Os parâmetros de entradas do método são o ponto de origem do eixo de coordenadas e um ponto que indica a direção do eixo cartesiano X.

4.3.3.10 Método `Polar`

O método Polar permite o cálculo de coordenadas na orientação polar. Os parâmetros de entrada são: um ponto de referência, o ângulo e a distância do vetor do deslocamento.

4.3.3.11 Método `PosicionaPosicao`

O método `posicionaPosicao` tem como objetivo o posicionamento do bloco posição em relação a barra criada. O posicionamento da posição deve ser tal que o bloco posição fique visível e bem posicionado em relação à barra e conseqüentemente necessite o mínimo possível de ajustes.

O método `posicionaPosicao` necessita do ponto inicial e final da barra para poder avaliar quando vale a pena posicionar, por exemplo, o texto acima ou abaixo da barra. Necessita ainda da escala do desenho, da orientação da cantoneira, do trusquino ou eixo principal de furação da barra e do ponto auxiliar de criação da barra.

4.3.3.12 Método `PosicionaTexto`

Assim como o método `posicionaPosicao` tem como objetivo o posicionamento do bloco posição, o método `posicionaTexto` tem como o objetivo o posicionamento do texto relacionado à barra.

Os parâmetros de entrada solicitados são os mesmos do método `posicionaPosicao`.

4.3.4 Classe `CdTcBancoDados`

A classe `CdTcBancoDados` está localizada no arquivo BancoDados.cs. Ela contém métodos para fazer a conexão com o banco de dados do aplicativo. Os métodos da classe `CdTcBancoDados` são: `incluirDados`, `excluirDados`, `atualizarDados` e `lerDados`.

A conexão com o banco de dados é realizada quando o aplicativo é iniciado. Já as instruções para realização de ações de inclusão, exclusão, modificação e leitura de dados são feitas dentro dos métodos através linguagem padrão de banco de dados SQL. As instruções SQL são armazenadas em variáveis do tipo texto e são então executadas.

4.3.4.1 Método `incluirDados`

Responsável pela inclusão de dados no banco de dados do aplicativo. É um método muito utilizado quando se faz a leitura dos dados do cálculo ou quando o projetista entra com os dados fornecidos por ele através das interfaces do programa.

O método `incluirDados` é sobrecarregado. Um método sobrecarregado é um método que possui o mesmo nome, mas pode possuir mais do que uma lista de parâmetros.

A primeira sobrecarga do método `incluirDados` recebe como parâmetro o objeto de conexão com o banco de dados, o nome da tabela que será modificada, o nome da coluna que será modificada e o valor que será adicionado.

A segunda sobrecarga é semelhante à primeira com exceção do último parâmetro. No caso da coluna ser preenchida com um texto, o valor não poderá ser um número e por conseqüência o parâmetro não poderá ser o mesmo da primeira sobrecarga. O último parâmetro da segunda sobrecarga é então um texto.

4.3.4.2 Método `excluirDados`

O método `excluirDados` é responsável pela exclusão de registros de tabelas do banco de dados. É utilizado principalmente quando alguma informação armazenada no banco de dados não vale mais, por exemplo, quando uma especificação utilizada anteriormente não será mais utilizada.

Assim como no método `incluirDados`, o método `excluirDados` é sobrecarregado. Os parâmetros de entrada são os mesmos do método `incluirDados` e a sobrecarga também é a mesma devido ao mesmo problema.

4.3.4.3 Método `atualizarDados`

O método `atualizarDados` é utilizado quando há necessidade de modificar dados existentes no banco de dados. Um exemplo da utilização desse método é quando há uma modificação nos dados do projeto ou na análise do projetista. Neste caso, as informações armazenadas no banco de dados necessitam ser modificadas.

Os parâmetros e as sobrecargas são os mesmos utilizados nos métodos anteriores.

4.3.4.4 Método `lerDados`

O método `lerDados` também é um método bastante utilizado pelo aplicativo desenvolvido. Sua finalidade é obter os dados armazenados no banco de dados para que eles possam ser utilizados por outros métodos com a finalidade de representar graficamente o detalhamento na tela da plataforma gráfica.

O método `lerDados` possui os parâmetros e sobrecargas como os métodos anteriores.

4.3.5 Classe `CdTcBarra`

A classe `CdTcBarra` armazena os métodos que são responsáveis pela representação gráfica da entidade barra na plataforma gráfica. Dentre os métodos presentes na classe CdTcBarra, podem-se destacar os métodos `CriaBarra` e `VesteBarra`.

4.3.5.1 Método CriaBarra

O método `CriaBarra` é o método que cria a barra a partir dos pontos iniciais e finais da barra. Possui nove parâmetros de entrada: aba, espessura, pinça, trusquino, orientação da Cantoneira, qualidade do aço, posição, escala e parafuso.

As variáveis aba, espessura, qualidade do aço e parafuso normalmente são fornecidas pelo cálculo. Já as outras variáveis são normalmente preenchidas pelo projetista responsável pelo desenho de acordo com o estudo da torre.

4.3.5.2 Método `VesteBarra`

O método `VesteBarra` permite a criação de várias barras simultâneamente. Os parâmetros de entrada do método são praticamente os mesmos do método `CriaBarra` com exceção da não existência dos parâmetros ponto inicial e ponto final. Os pontos iniciais e finais das barras criadas são provenientes das linhas de furação escolhidas pelo usuário.

4.3.6 Classe `CdTcPrompt`

A classe `CdTcPrompt` tem o objetivo de fazer a ligação entre o aplicativo e a linha de comando do AutoCAD. Através da linha de comandos do AutoCAD o usuário poderá entrar com informações ou receber informações sobre os comandos do aplicativo. A linha de comando do AutoCAD é ilustrada na Figura 4-15.

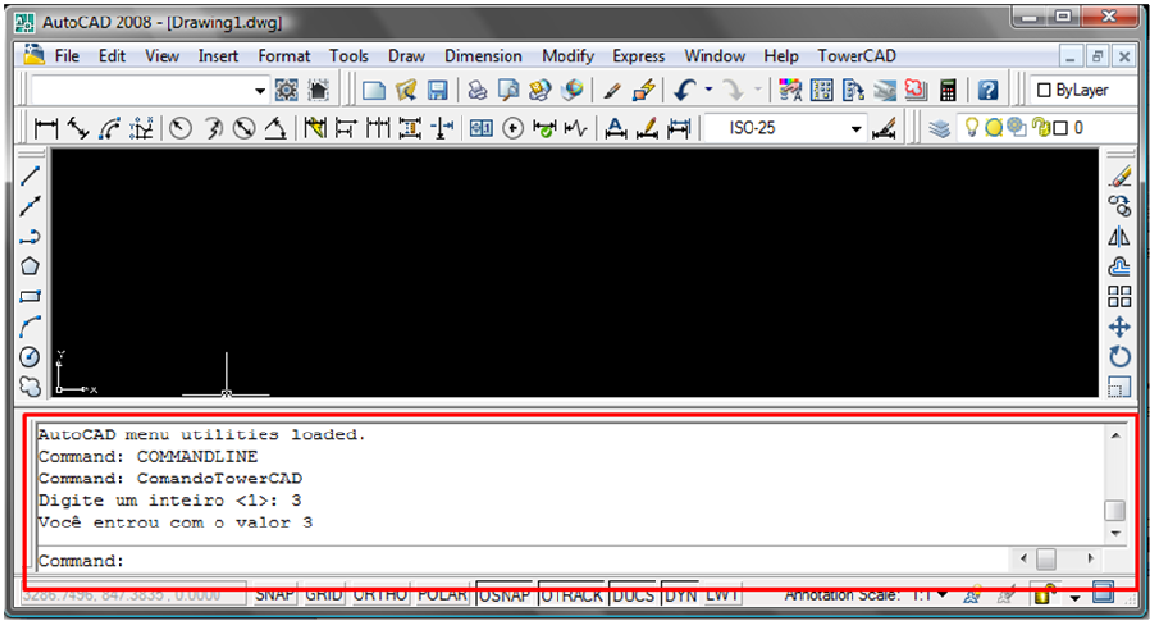


Figura 4-15 - Linha de comandos do AutoCAD

A classe `CdTcPrompt` possui dois métodos chamados `ReceberPrompt` e `EscreverPrompt`. Esses métodos possuem sobrecarga para receber como parâmetros números inteiros, reais e texto.

4.3.6.1 Método `EscreverPrompt`

O método `EscreverPrompt` é responsável por escrever na linha de comandos alguma informação fornecida pelo aplicativo. As sobrecargas do método variam de acordo com o tipo de informação que será fornecida pelo aplicativo.

Existem as sobrecargas para valores do tipo inteiro, do tipo real e para textos. Os parâmetros solicitados pelas funções são o tipo de informação, que é de acordo com a sobrecarga, e o texto que estará localizado antes da informação fornecida pelo aplicativo.

4.3.6.2 Método `ReceberPrompt`

O método `ReceberPrompt` é responsável por receber as informações digitadas pelo usuário na linha de comandos do AutoCAD. O método é sobrecarregado de acordo com tipo de informação que o aplicativo deseja obter do usuário.

Os parâmetros solicitados são o tipo de informação, de acordo com a sobrecarga, e o possível texto que será impresso na tela para pedir ao usuário que digite a informação.

4.3.7 Classe `CdTcOutros`

A classe `CdTcOutros` é uma classe que possui métodos pouco utilizados pelo aplicativo, mas que podem ser úteis durante as próximas etapas de elaboração do mesmo. Possui dois métodos: o método `SelecionaEntidades` e o método `leAtributosBloco`.

4.3.7.1 Método `SelecionaEntidades`

O método `SelecionaEntidade` permite que o usuário escolha uma lista de entidades no AutoCAD e que a partir dessa escolha, o aplicativo possa obter todas as informações sobre as entidades selecionadas pelo usuário. Dados como a cor, *layer* e identificador podem ser rapidamente localizados e trabalhados em futuros algoritmos.

4.3.7.2 Método `leAtributosBloco`

O método `leAtributosBloco` pede como parâmetro a referência do bloco a ser analisado e retorna todos os atributos que estão relacionados com ele.

4.3.8 Classe `CdTcDadosBarra`

A classe `CdTcDadosBarra`, assim como as próximas classes que serão descritas neste trabalho, são classes derivadas da classe Windows Form do Framework .NET. Permitem uma interface com o usuário mais fácil e amigável.

A classe `CdTcDadosBarra` tem a finalidade de fazer a interface gráfica com o usuário do comando `CBarra`. O formulário correspondente da classe `CdTcDadosBarra` pode ser visualizado na Figura 4-16.

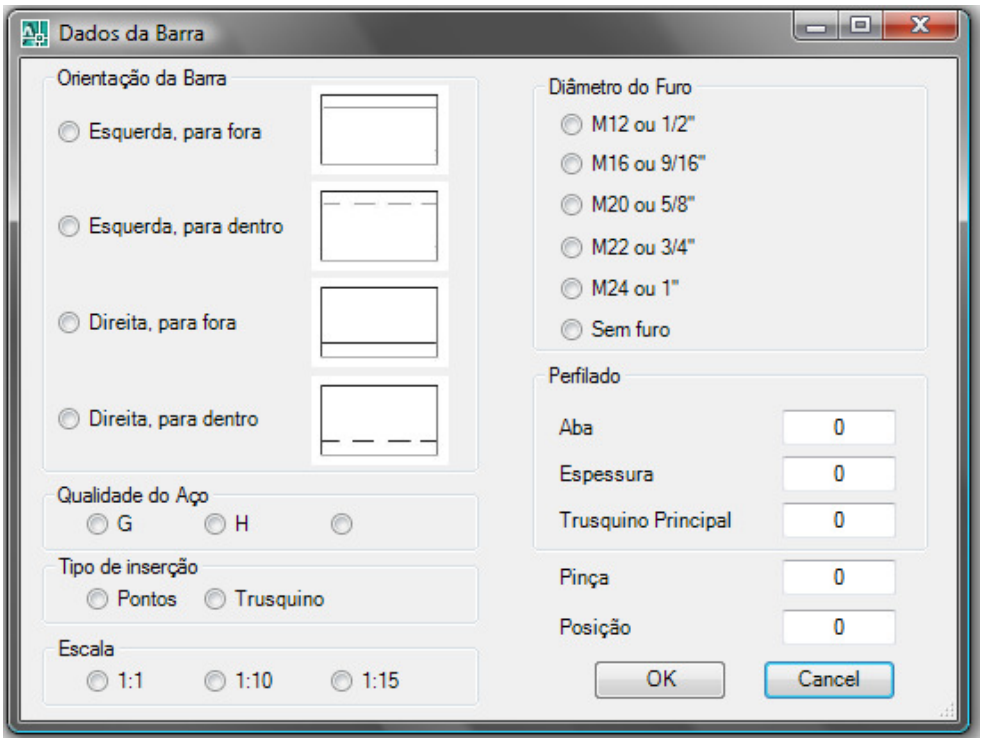


Figura 4-16 – Classe `CdTcDadosBarra`

A orientação da barra é definida a partir da escolha de uma das opções. O controle utilizado na interface chamada *RadioButton* permite que apenas uma das opções de orientação de barra seja escolhida. Isso é coerente com essa entrada de informação visto que uma barra não pode ter mais de um tipo de orientação. Uma figura foi colocada ao lado de cada opção para o entendimento do usuário fique mais fácil

As entradas do usuário qualidade do aço, tipo de inserção, escala e diâmetro do furo também utilizam o mesmo tipo de controle da orientação da barra devido ao fato de só poder receber uma das opções.

As outras entradas do usuário são feitas pelo controle chamado *TextBox*. Os controles *TextBox* permitem que o usuário digite valores ou textos. Um controle para assegurar que o

usuário digitou com valor coerente com o que foi pedido é feito no arquivo de código de classe através do uso de exceções. Os campos aba, trusquino, espessura, pinça e posição utilizam esses controles.

4.3.9 Classe `CdTcDadosGeometria`

A classe `CdTcDadosGeometria` cria o objeto de interface do comando `CGeometria` (criado neste trabalho e descrito na seção 4.4.3). A visualização do formulário da classe pode ser feita na Figura 4-17.

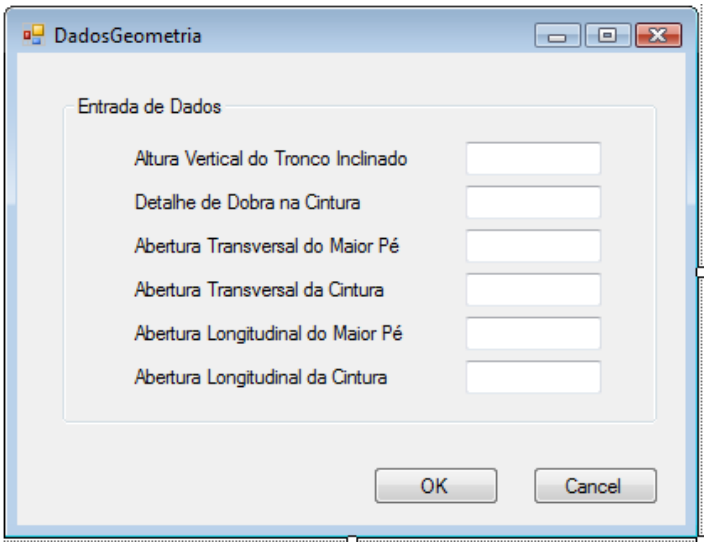


Figura 4-17 - Classe `CdTcDadosGeometria`

Os controles para entrada de parâmetros pelo usuário nessa classe são todos do tipo *TextBox*. O controle para assegurar que o usuário digitou valores coerentes é feito no arquivo de código da classe.

4.3.10 Classe `CdTcNormaParafuso`

A classe `CdTcNormaParafuso` cria o objeto de interface que permite que o usuário escolha qual das normas sobre parafusos ele quer acrescentar. O formulário da classe pode ser visualizado na Figura 4-18.

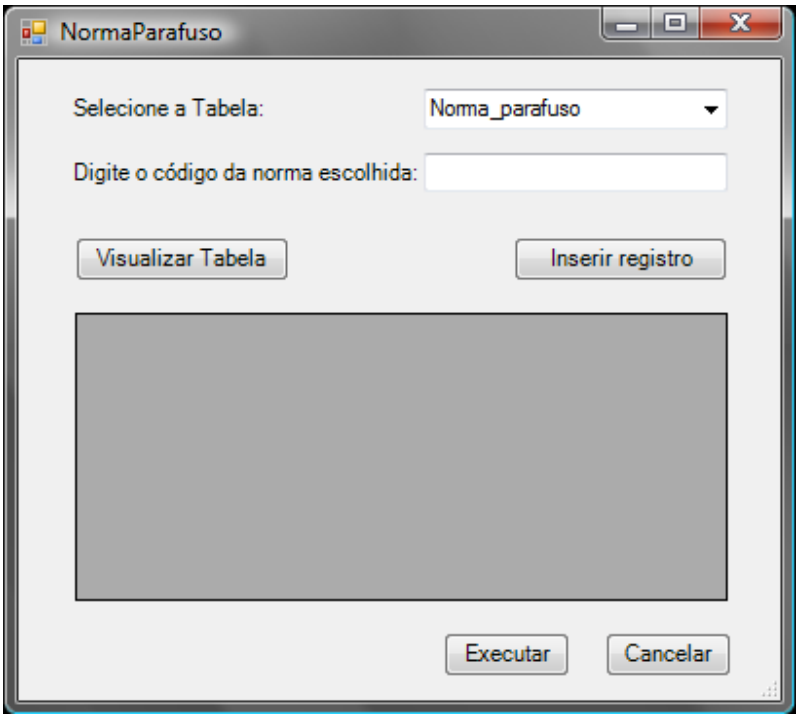


Figura 4-18 - Classe `CdTcNormaParafuso`

Essa classe possui alguns controles, na interface, diferentes das outras. O controle representado por um grande retângulo é o *dataGridView*. Esse é um controle que permite um *link* dinâmico do AutoCAD com o banco de dados do aplicativo para que as tabelas possam ser visualizadas. As informações podem ser vistas no controle logo após serem salvas no banco de dados.

O controle ao lado do texto "Selecione a Tabela:" é o controle chamado *comboBox*. O controle *comboBox* permite que o usuário escolha, em uma lista flutuante, qual o nome da tabela que ele deseja modificar.

4.3.11 Classe `CdTcIncluirNormaParafuso`

Essa classe, assim como as próximas, é uma interface para inclusão de dados em tabelas do banco de dados do aplicativo. São como formulários para inclusão de dados. Elas possuem como parâmetros de entrada os campos que devem ser preenchidos em cada registro.

A classe `CdTcIncluirNormaParafuso` é a classe que cria o objeto de interface para inclusão de registros na tabela `Norma_parafuso` que está descrita na seção 5.3.10. O formulário da classe pode ser visualizado na Figura 4-19.

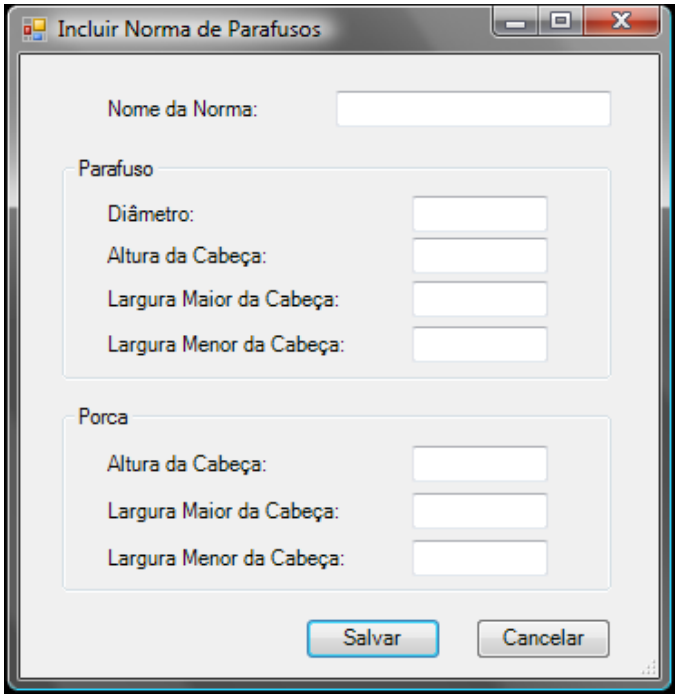


Figura 4-19 - Classe `CdTcIncluirNormaParafuso`

Só possui controles do tipo *textBox*, assim as próximas classes.

4.3.12 Classe `CdTcIncluirDistanciaParafuso`

A classe `CdTcIncluirDistanciaParafuso` é a classe que cria o objeto de interface para inclusão de registros na tabela `Distancia_parafuso` que está descrita na seção 5.3.10. O formulário da classe pode ser visualizado na Figura 4-20

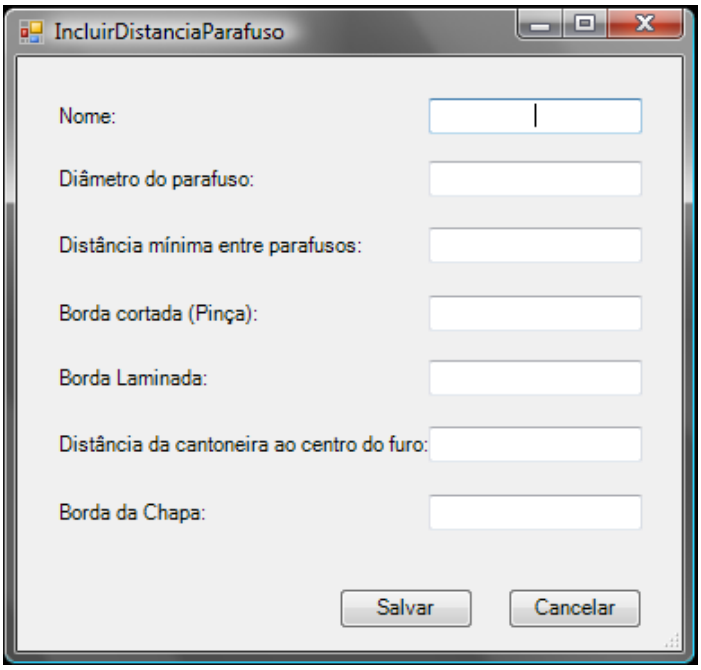


Figura 4-20 - Classe `CdTcIncluirDistanciaParafuso`

4.3.13 Classe `CdTcIncluirComprimentoParafuso`

A classe `CdTcIncluirComprimentoParafuso` é a classe que cria o objeto de interface para inclusão de registros na tabela `Comprimento_parafuso` que está descrita na seção 5.3.10. O formulário da classe pode ser visualizado na Figura 4-21.

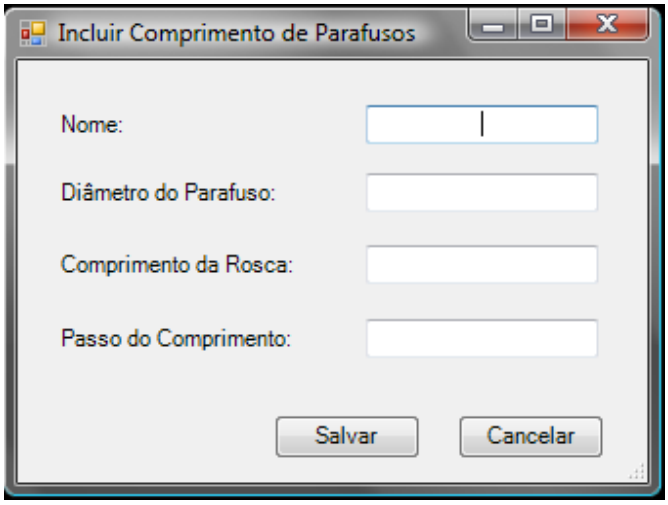


Figura 4-21 - Classe `CdTcIncluirComprimentoParafuso`

4.4 Comandos desenvolvidos

Os comandos desenvolvidos para o aplicativo TowerCAD rodam no AutoCAD como se fossem comandos nativos da plataforma gráfica e é através deles que todas as ações, desenvolvidas pelo aplicativo, são executadas.

Da mesma for que o AutoCAD os comandos podem ser acessados pela linha de comandos ao ser digitado o nome do comando, pela barra de ferramentas criada para aplicativo ou ainda pelo menu "pulldown" criado no AutoCAD. A Figura 4-22 mostra o ambiente do AutoCAD customizado pelo aplicativo TowerCAD com destaque para o menu "pulldown" e para barra de ferramentas criados.

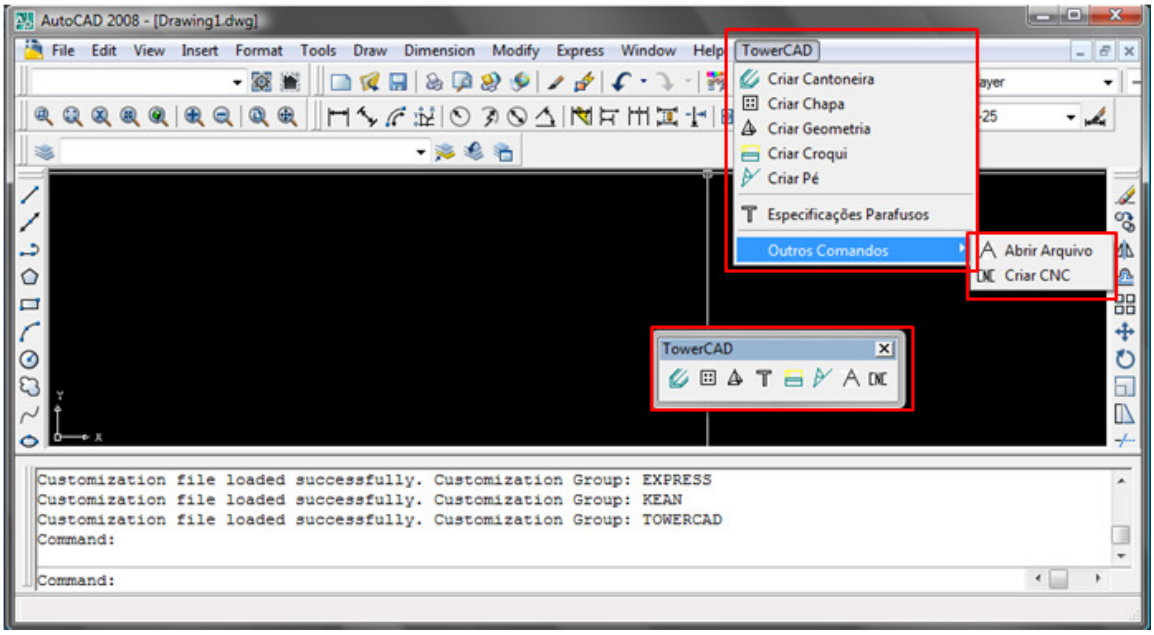


Figura 4-22 - Ambiente do AutoCAD customizado pelo TowerCAD

Durante o acompanhamento do trabalho dos projetistas, foi constatado que muitas atividades executadas eram repetitivas e facilmente automatizáveis através das tecnologias aprendidas. Para aumentar a produtividade e automação do processo, foram criados então alguns comandos com rotinas, que evitam a ocorrência de erros e diminuem o tempo de execução de tarefas.

4.4.1 Comando `CBarra`

O comando `CBarra` cria uma barra a partir de uma linha ou de dois pontos definidos pelo usuário utilizando a interface gráfica amigável com o usuário mostrada na Figura 4-22.

O comando permite que o usuário escolha a orientação, o tipo de parafuso, a qualidade do aço, a escala, a aba, o trusquino, a espessura, a pinça e a posição da barra. Os valores são todos inseridos pelo usuário através da interface gráfica amigável. Os valores digitados pelo usuário, na primeira vez que utiliza o comando, são armazenados e são carregados quando o usuário invoca o comando mais uma vez. Assim, muitas vezes o usuário não necessita digitar as mesmas informações várias vezes o que aumenta a rapidez de execução.

O usuário tem a opção ainda de escolher se ele quer criar a barra a partir de um trusquino (linha de eixo de furação), que seria representado no AutoCAD por um objeto linha ou ainda se ele deseja criar a barra a partir da seleção de dois pontos. A seleção de criação a partir do trusquino permite que o usuário crie múltiplas barras ao mesmo tempo se ele selecionar várias linhas. A posição de cada barra vai aumentando de acordo com a ordem de seleção das linhas. A visualização da execução do comando no modo de criação a partir do trusquino é apresentada na Figura 4-23.

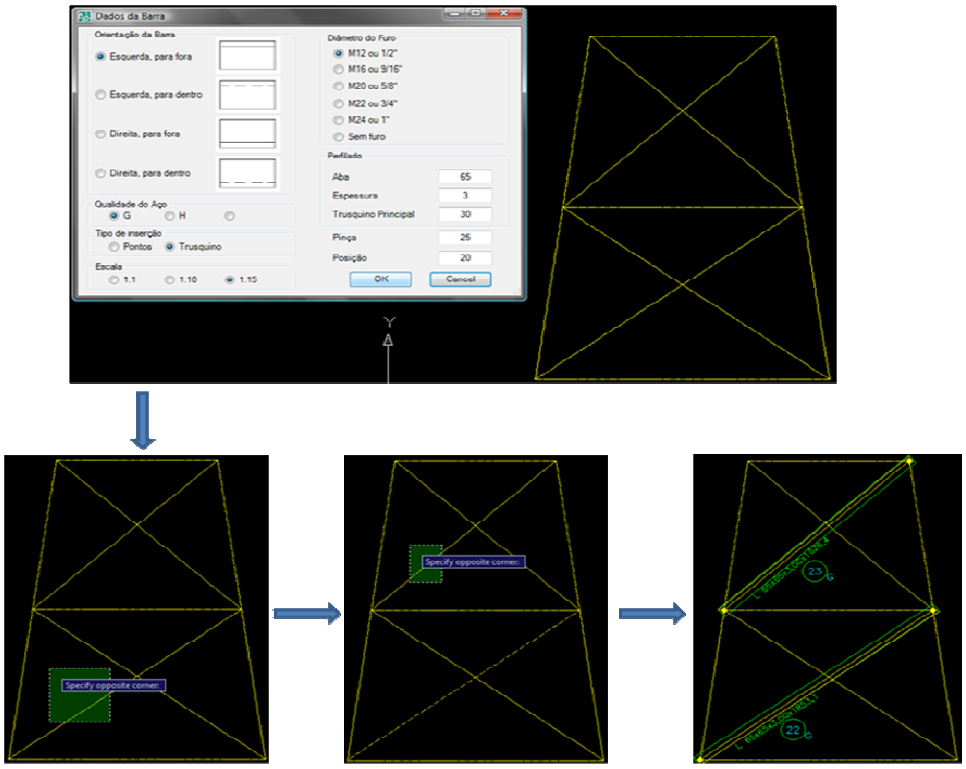


Figura 4-23 - O Comando `CBarra`

Esse comando pode ser muito produtivo quando o projetista faz o estudo da torres a partir dos eixos de furação. Ao "vestir" a torre rapidamente, grande produtividade pode ser alcançada. Tarefas repetitivas e extremamente mecânicas deixam de ser executadas.

4.4.2 Comando `CPlaca`

O comando `CPlaca` cria uma placa a partir da seleção de parafusos feita pelo usuário. Foi observado que a criação de placas é uma tarefa que despende um tempo razoável que poderia ser facilmente diminuído.

Existe um filtro de seleção que permite que o comando selecione somente as entidades parafuso então, mesmo que o usuário tente selecionar outras entidades do AutoCAD como, por exemplo, linhas e textos, o comando não as selecionará. Somente serão selecionados os parafusos e a partir deles será criada a chapa.

As distâncias mínimas do centro dos parafusos até a borda da chapa variam com o diâmetro dos parafusos e são obtidas da tabela `distancia_parafuso` (criada neste trabalho e descrita na seção 5.3.10) no campo borda_chapa.

Na Figura 4-24 pode-se observar a sequência de execução do comando. Primeiramente, pede-se ao usuário que selecione os parafusos que estarão na chapa a ser criada. Após a seleção, o comando é automaticamente executado e a chapa é desenhada na tela.

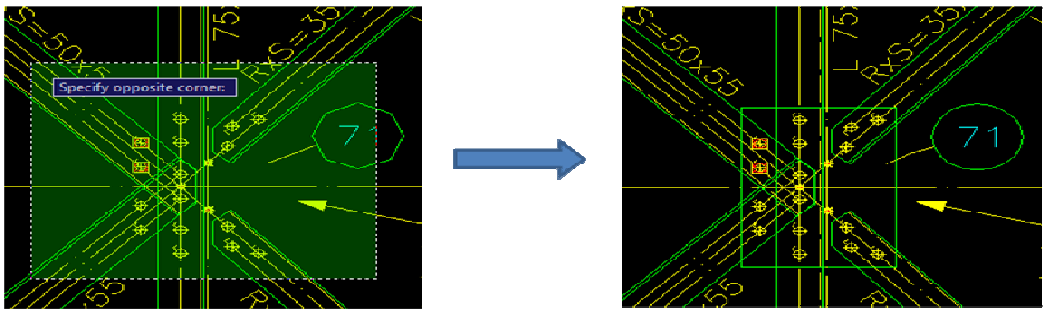


Figura 4-24 - O Comando `CPlaca`

4.4.3 Comando `CGeometria`

O comando `CGeometria` tem a finalidade de fazer o cálculo automático da geometria da "pirâmide" da torre através do método `CriaGeometria` da classe CdTcAuxiliar descrito na seção 4.3.3.1 e ainda representar graficamente o resultado na tela. A interface gráfica do comando é um objeto da classe `CdTcDadosGeometria` descrita na seção 4.3.9.

A necessidade de criação desse comando ocorreu a partir da observação do modo de trabalho dos projetistas. Os cálculos são feitos à mão e depois disso os resultados são digitados na tela. O fato de o cálculo ser feito a mão aumenta a possibilidade de ocorrência de erros e o fato dos resultados serem digitados na tela significa que mesmo com o cálculo correto, a sua representação na tela pode estar errada pela interferência humana ao digitar os dados.

O cálculo da geometria da "pirâmide" da torre pode ser considerado um dos estudos mais importantes feitos antes do início do detalhamento da torre. Se o cálculo da geometria estiver incorreto, todas as dimensões da parte inclinada da torre estarão incorretas.

A representação da geometria da "pirâmide" da torre nos desenhos de detalhamento está na Figura 4-25.

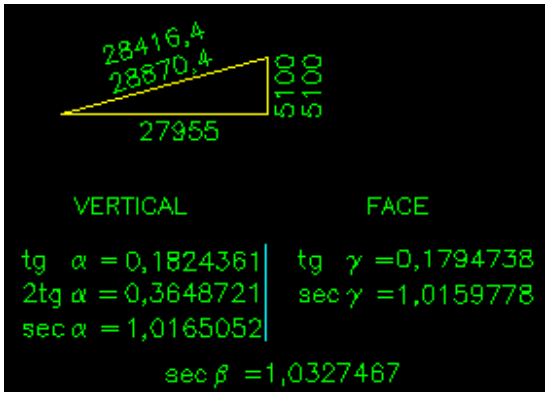


Figura 4-25 - Representação da geometria nos desenhos de detalhamento

4.4.4 Comando `CCroqui`

O comando `CCroqui` gera na tela os croquis das peças para fabricação. É um dos subprodutos gerados pelo sistema TowerCAD. Os croquis detalham todas as distâncias necessárias para fabricar a peça. A Figura 4-26 permite visualizar o croqui de uma peça com apenas um trusquino e que é ligada a outras peças por um parafuso.

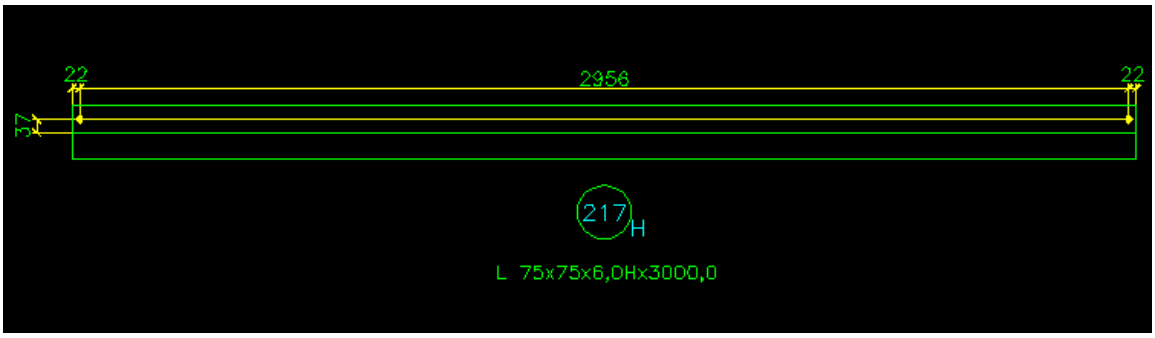


Figura 4-26 - Croqui de uma peça para fabricação

4.4.5 Comando `CPe`

O comando `CPe` é responsável pelo detalhamento dos pés das estruturas. A partir dos dados fornecidos pelo cálculo dos perfis utilizados e ainda de alguns dados fornecidos pelo

projetista de acordo com o estudo realizado por ele, o comando permite o desenho praticamente completo de um pé da estrutura. É o comando que permitiu o maior aumento de produtividade no processo de detalhamento de torres.

O desenho de detalhamento completo de um pé de um metro e meio de altura de uma torre pode ser visualizado na Figura 4-27.

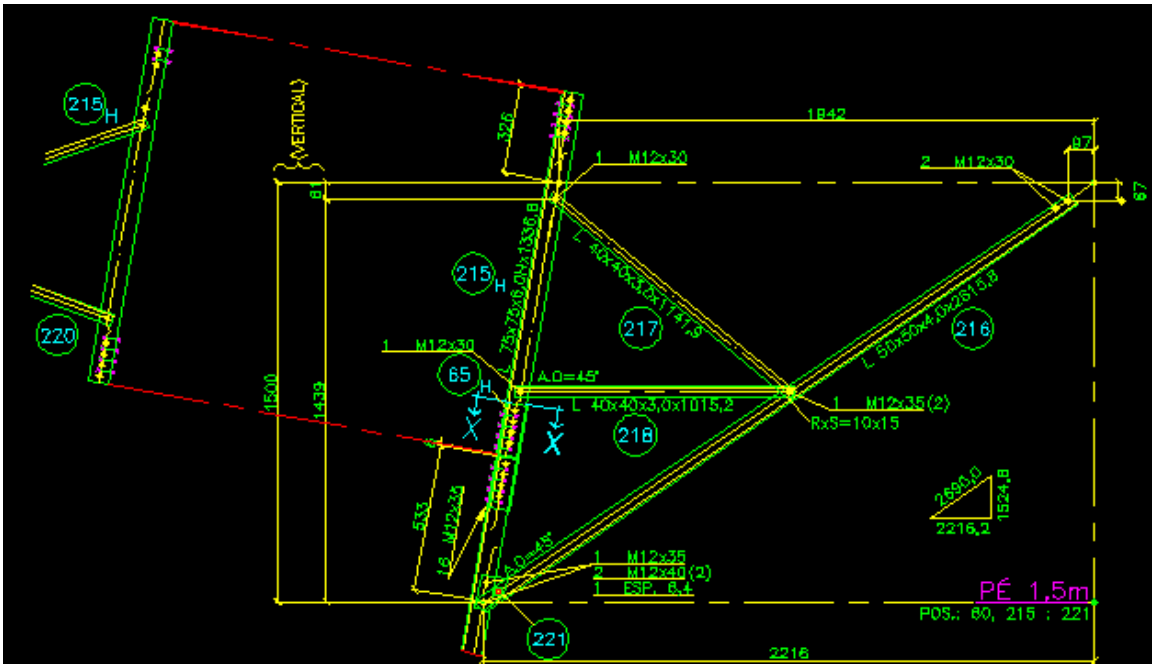
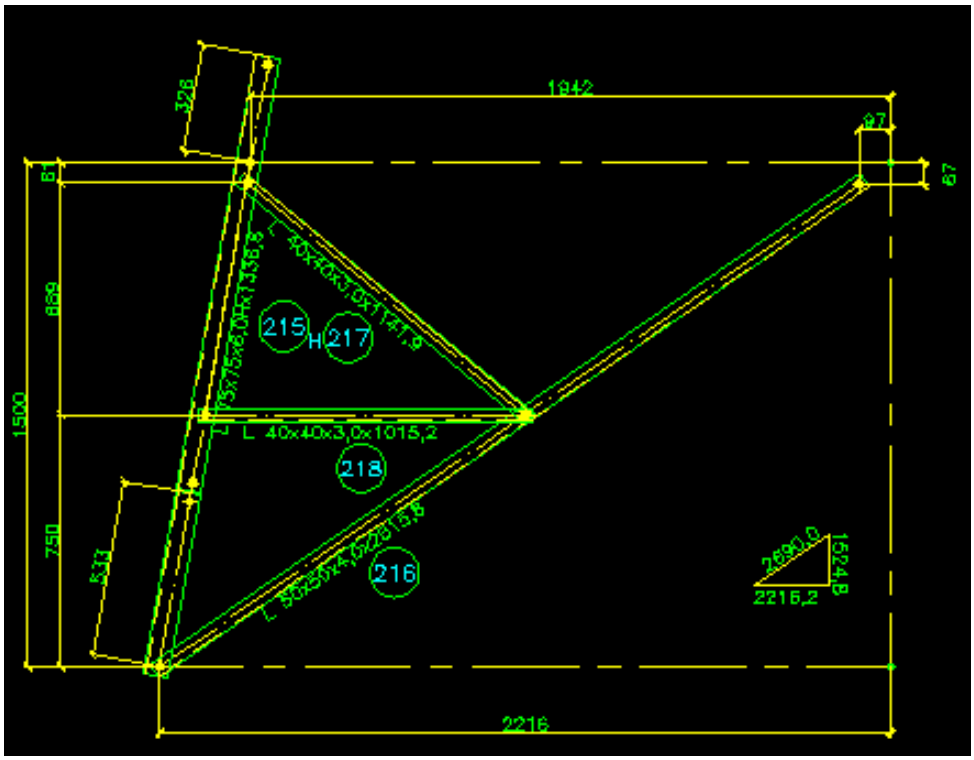


Figura 4-27 - Detalhamento completo de um pé de uma torre

A mesma estrutura detalhada automaticamente, a partir do comando `CPe` pode ser visualizada na Figura 4-27.



4-28 - Desenho de detalhamento de um pé gerado automaticamente pelo `CPe`

4.4.6 Comando `PParafuso`

O comando `PParafuso` fornece uma interface gráfica para que os projetistas entrem com os dados dos parafusos que serão utilizados no projeto.

Utilizando como base a interface gráfica criada pela classe `CdTcNormaParafuso`, o usuário pode visualizar quais as normas que estão armazenadas no banco de dados e decidir se deseja incluir uma nova norma. As normas mais utilizadas pelos projetistas de torres metálicas para distâncias mínimas entre parafusos são as normas da AISC(2006) e ABNT(2008).

A visualização da interface pode ser feita na Figura 4-18. A visualização das tabelas do banco de dados ocorre em tempo real.

Ao escolher uma norma pelo código e clicar no botão executar, o projetista informa ao aplicativo que aquelas serão as informações que deverão utilizadas ao longo do projeto desenvolvido. Devem ser escolhidas normas para as três especificações de parafuso existentes. A Figura 4-29 mostra a norma de código número 5, pertencente à tabela `Norma_parafuso` descrita na seção 5.3.10, sendo escolhida pelo usuário do aplicativo.

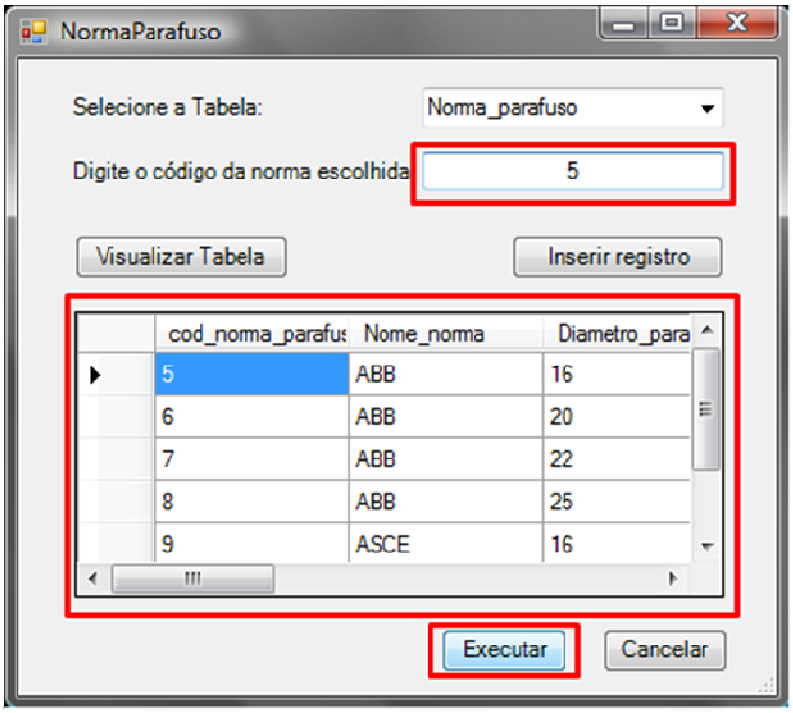


Figura 4-29 - Escolha da Norma de Parafusos

Ao observar que a norma a ser utilizada não está presente no banco de dados, o usuário pode acrescentar uma nova norma clicando no botão "Inserir registro". A interface que aparecerá irá depender do conteúdo escolhido no campo "Selecione a Tabela:". Esse conteúdo poderá ser objeto das classes `CdTcIncluirComprimentoParafuso`, `CdTcIncluirDistanciaParafuso` ou `CdTcIncluirNormaParafuso`. A Figura 4-30 permite visualizar a escolha e as possíveis janelas que aparecerão.

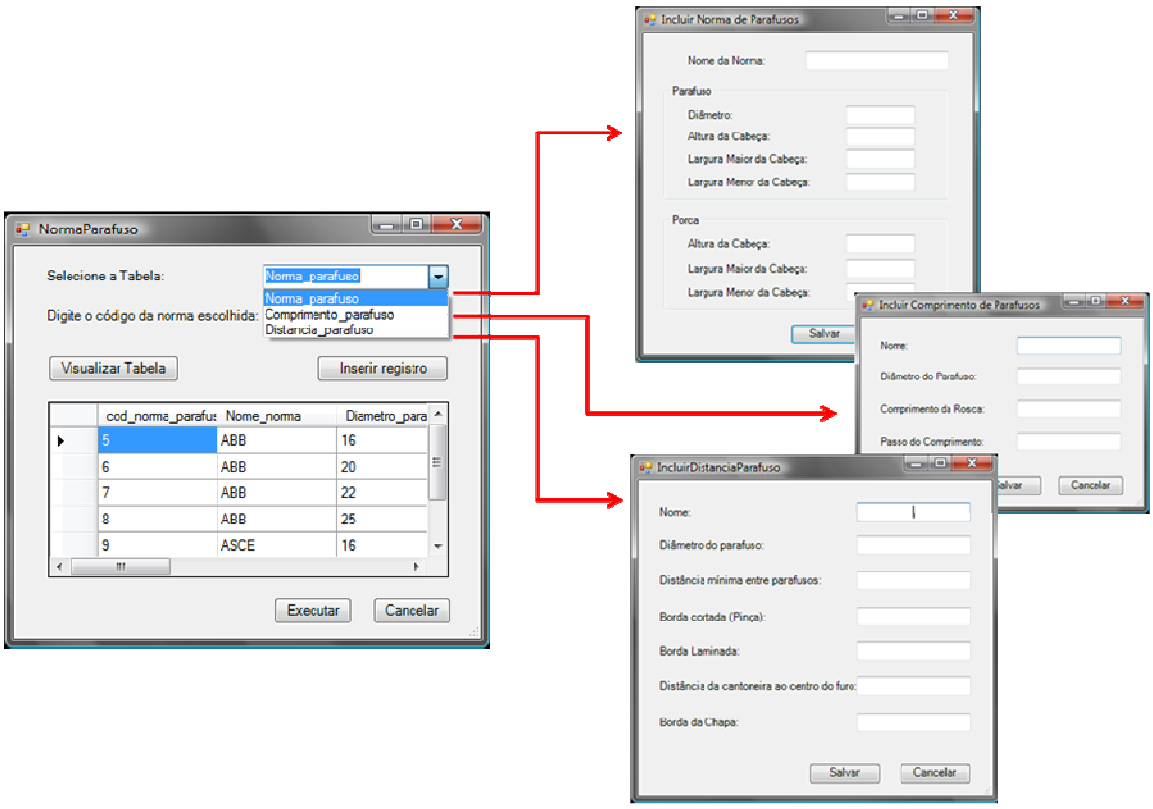


Figura 4-30 - Inclusão de Norma

Depois de acrescentada, a norma poderá ser visualizada e possuirá um código para ser escolhida pelo usuário.

4.4.7 Comando `LerArquivo`

O comando `lerArquivo` é responsável por ler um arquivo, com extensão `.dat`. Esse arquivo possui informações sobre resultados das etapas de entrada de dados, análise e dimensionamento. O referido comando também é responsável por armazenar no banco de dados do aplicativo os dados presentes no arquivo lido. Faz, portanto, a integração entre as etapas anteriores e a etapa de detalhamento do processo produtivo de torres metálicas.

Ao ser chamado, o comando abre uma janela padrão do Windows para que o usuário possa escolher o arquivo conforme mostrado Figura 4-31.

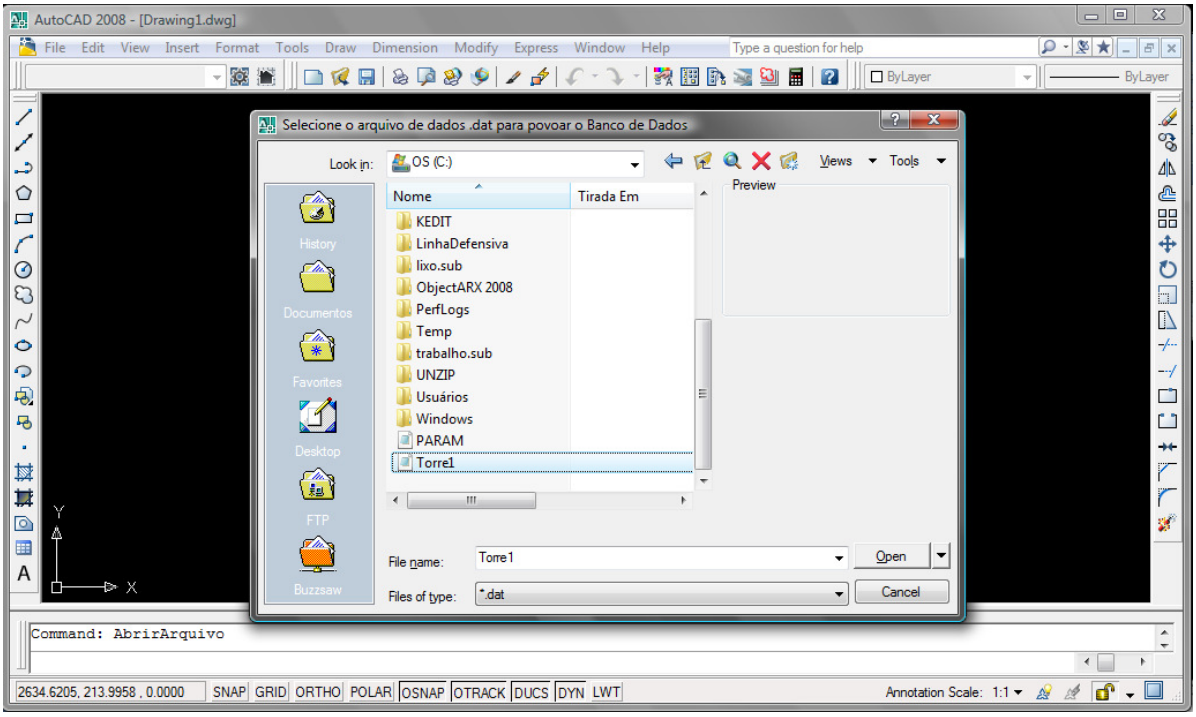


Figura 4-31 - Seleção do arquivo para o povoamento do banco de dados

O nome do arquivo será tratado pelo aplicativo como o nome da torre a ser projetada. Caso exista uma torre com o mesmo nome, armazenada no banco de dados, o aplicativo se certifica através da interface com o usuário que é realmente a mesma torre.

Na Figura 4-33, é mostrada uma configuração possível de arquivo .DAT que foi implementado pelo programa e que possui informações provenientes das etapas anteriores à etapa de detalhamento. Dentre as informações obtidas, podem-se destacar: o nome da barra, os nós que ela interliga, o código do perfil, o comprimento, a quantidade e o tipo de parafuso utilizado.

```
F1     112     1     6 1001     2    12
P15     46     2     5 3020     1    12
1501P   24     3     7  530     1    12
1502P   24     3     8  480     1    12
1503P   24     7    11  510     1    12
1504P   24     7    12  430     1    12
1505P   24    11    15  460     1    12
1506P   24    11    16  380     1    12
1507P   24    15    19  410     1    12
1508P   24    15    20  330     1    12
1509P   24    19    23  260     1    12
1510P   24    19    24  280     1    12
```

Figura 4-32 – Configuração de um arquivo .DAT modelo

4.4.8 Comando `GerarArquivo`

O comando Gerar Arquivo, para a fase CAM do processo, é responsável por criar o arquivo que é utilizado na programação dos equipamentos responsáveis pela fabricação das peças. Faz a integração entre a etapa de detalhamento e a etapa de fabricação do processo produtivo de torres metálicas.

A formatação dos arquivos gerados depende do fabricante que contrata o serviço. A formatação do arquivo mostrado na Figura 4-33 é de um fabricante específico e foi implementada no programa. Todos os dados necessários para fabricação devem estar transcritos no arquivo: o que vai estar escrito na peça, nome da estrutura, recortes, dobras, aço utilizado, posicionamento dos furos, diâmetro dos furos e dimensões básicas da peça.

```
TORRE T1

POSICAO
217

A 572

FURO 175
X  22
Y  20
X  2978
Y  20

COMPRIMENTO
3000

ABA
75

ESPESSURA
6
```

Figura 4-33 – Formatação de um arquivo CAM para controle numérico gerado pelo comando GerarArquivo

5. O BANCO DE DADOS

Embora independente, o banco de dados é outro componente do sistema CAD desenvolvido neste trabalho. Nesse capítulo é apresentada uma visão geral do banco de dados com suas tabelas, relacionamento entre tabelas e uma breve descrição de como foi elaborada a estrutura e o povoamento do banco de dados. A tecnologia ADO.NET utilizada permitiu executar as tarefas relacionadas com o acesso e a manutenção de dados através de suas classes. Os componentes ADO.NET foram projetados para tratar o acesso e a manipulação dos dados facilitando a realização de tarefas como manipulação das tabelas para inclusão/exclusão/atualização e leitura de dados.

5.1 Visão Geral

O objetivo da construção de um banco de dados neste trabalho é garantir o gerenciamento e a integridade de dados ao longo do processo de fabricação de estruturas de torres e permitir o intercâmbio de informações necessárias às diversas etapas (CAD/CAE/CAM), garantindo assim a confiabilidade do sistema global conforme Gontijo (2009).

A desvinculação das etapas do processo de fabricação de torres, que são muitas vezes desenvolvidas independentemente por equipes diferentes, ocasiona, na maioria das vezes, a ocorrência de erros e problemas no processo de transferência de dados. A falta de integração entre as atividades de um processo tem como conseqüências interrupções indesejadas, o aumento de custos e o comprometimento da qualidade. A possibilidade de erros aumenta substancialmente quando o fator humano manipula os dados, altera formatações e realiza conversões de unidades ao transferir dados de uma plataforma para a outra.

Um banco de dados externo possibilita a vinculação das etapas a partir do armazenamento de todas as informações necessárias de cada uma delas. O banco de dados representa um sistema de integração consistente, completo e correto, que garante um alto grau de confiabilidade ao processo, a qualidade do produto e a eficiência das soluções adotadas.

Dados referentes ao detalhamento podem ser retirados do banco de dados externo criado e posteriormente ser transmitidos aos sistemas CAM. O banco de dados pode ainda fornecer diversas informações sobre o projeto como, por exemplo, a lista de materiais e outros relatórios.

No Banco de Dados são armazenados todos os dados necessários para a representação gráfica. O objetivo do uso de um banco de dados externo é permitir que os dados armazenados estejam em uma plataforma independente da plataforma gráfica, o que permite uma maior flexibilidade e portabilidade sistema computacional desenvolvido. Foi criada uma estrutura complexa de tabelas e relacionamentos para armazenar os dados. Uma parte dessa estrutura é exemplificada na Figura 5-1.

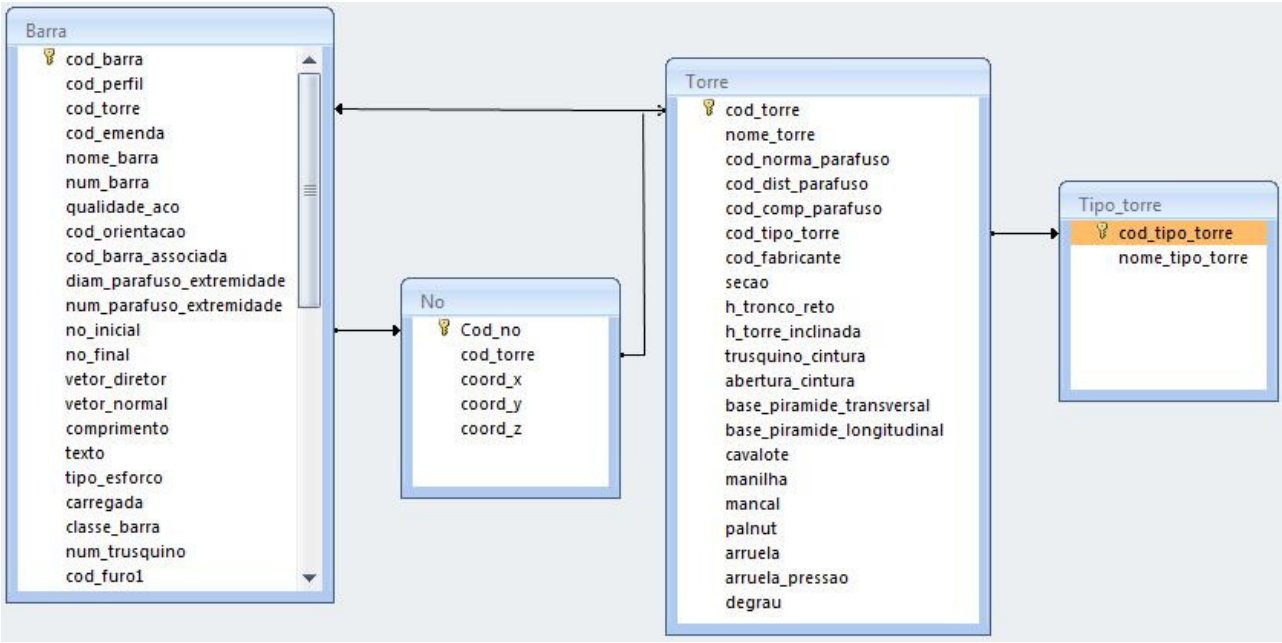


Figura 5-1 - Parte da estrutura do Banco de Dados

Para o acesso aos dados, a conexão com o banco de dados é aberta assim que o aplicativo é carregado no AutoCAD e fechada quando o aplicativo é descarregado ou quando o AutoCAD é fechado como mostra o código abaixo.

```
    public class Initialization : IExtensionApplication
    {
        static OleDbConnection dataConnection = new OleDbConnection();

        static public OleDbConnection getConnection()
        {
            return dataConnection;
        }

        public void Initialize() //Inicialização do programa..
        {
            dataConnection.ConnectionString =
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source=C:\\BancoDados.accdb";
            dataConnection.Open();
        }

        public void Terminate() //Término do programa...
        {
            dataConnection.Close();
        }
    }
```

5.2 A Construção da Estrutura do Banco de Dados

O primeiro passo para montar a configuração do banco de dados foi estudar e analisar estruturas de classes de trabalhos anteriores. Em Magalhães (2002), foi criada uma estrutura de classes para detalhar torres metálicas e em Leite (2002) foi criada uma estrutura para detalhar estruturas de concreto pré-moldado Essas estruturas de classes serviram de base para o desenvolvimento da estrutura desse trabalho.

Outra atividade importante para definir a estrutura do banco de dados para este trabalho foi o acompanhamento participativo do processo de produção de desenhos de detalhamento na prática, como é feito atualmente por projetistas. Para tal, foram executados detalhamentos de algumas torres com a supervisão de projetistas experientes. Essa experiência, permitiu levantar e analisar outros dados importantes para o detalhamento de estruturas de torres metálicas, definir os tipos de informações/dados que devem ser armazenadas e persistidas ao longo da elaboração do projeto e em especial para definir critérios de automação desta etapa.

Os dados foram então agrupados em tabelas de acordo com o relacionamento entre si. Segundo Battisti (2009), assuntos não devem ser misturados em uma mesma tabela. As tabelas foram, por isso, elaboradas de modo a terem a capacidade de armazenar dados que tenham relação entre si e sejam necessários para exibir na plataforma gráfica o produto final do detalhamento. O modelo lógico de banco de dados utilizado é o representado da maneira relacional.

Após definidas as tabelas necessárias e os campos que a compõe, foram analisados e definidos quais deles seriam utilizados como chave primária. Os campos definidos como chave primária são únicos nas tabelas e servem para fazer as ligações entre as tabelas.

Na sequência, foram elaborados relacionamentos entre as tabelas. Em alguns casos, quando os relacionamentos entre as tabelas eram do tipo `Vários para vários`, foi necessária a criação de tabelas auxiliares para fazer a ligação. Foram criados ainda relacionamentos do tipo `Um para Vários`.

Finalmente, foi feito o refinamento da estrutura do Banco de Dados através do processo chamado normatização. Segundo Battisti (2009), a normatização tem como objetivo evitar os problemas provocados por falhas no projeto de Banco de Dados, bem como eliminar a "mistura de assuntos" e as correspondentes repetições desnecessárias de dados. Quando o modelo relacional de dados é utilizado, deve-se evitar a mistura de assuntos em uma mesma tabela.

Foi observado que algumas tabelas anteriormente criadas poderiam ser unificadas com outras de modo a eliminar repetições desnecessárias de dados e outras poderiam ser criadas para evitar a mistura de assuntos.

A tabela `Legenda`, por exemplo, incorporou a tabela `Detalhes_notas`, definida inicialmente, pois foi constatado que os campos das duas tabelas poderiam ser agrupados em uma só tabela sem perda de qualidade da estrutura de banco de dados. A tabela `Detalhes_notas` foi então excluída da estrutura de banco de dados e a tabela `Legenda` armazenou os campos anteriormente pertencentes à tabela `Detalhes_notas`.

5.3 A Elaboração das Tabelas

O princípio utilizado para a elaboração das tabelas foi analisar todas as informações necessárias para a elaboração dos desenhos de detalhamento e organizar essas informações estabelecendo ainda as possíveis relações entre elas e agrupando-as em tabelas.

Algumas tabelas foram organizadas de modo a representar os elementos presentes no detalhamento de torres metálicas e contém todos os dados necessários para a representação gráfica dos mesmos. Outras foram criadas para armazenar as especificações de como detalhar esses elementos.

Dentre as tabelas criadas a partir dos elementos presentes no detalhamento de torres, podem-se destacar as tabelas: `Barra`, `Furo`, `Ligação`, `Emenda` e `Chapa`. Na Figura 5-2, podem-se visualizar esses elementos presentes em um desenho de detalhamento.

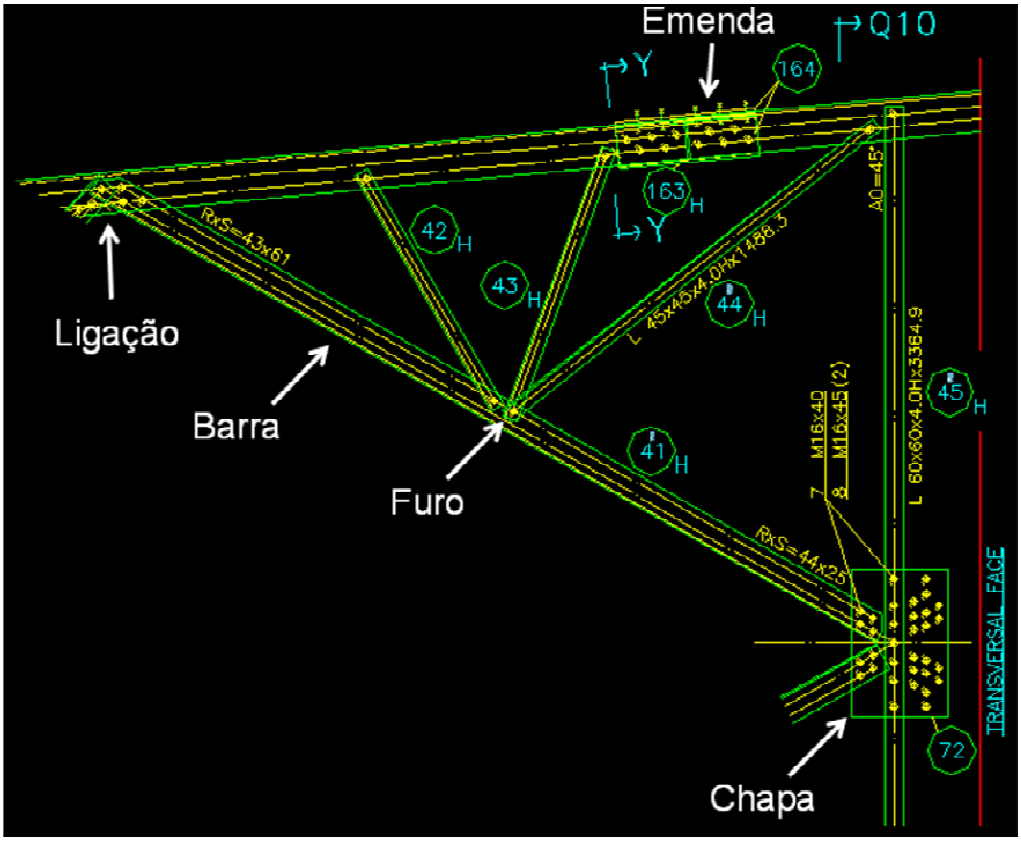


Figura 5-2 - Elementos do detalhamento

A tabela criada `Torre` pode ser considerada a tabela "mãe" da estrutura de banco de dados e possui relação com muitas das tabelas da estrutura. A tabela `Tipo_torre` indica qual é o tipo da torre a ser desenvolvida.

Para caracterizar os parafusos foram criadas três tabelas: `Norma_parafuso`, `Comprimento_parafuso e Distancia_parafuso`. As tabelas não puderam ser unificadas porque existem casos, em que informações de uma das tabelas podem ser relacionadas com mais de um tipo de informação das outras tabelas.

Para desenhar os formatos nos quais os desenhos serão entregues foram criadas as tabelas: `Desenho, Formato, Nota e Legenda`.

Foram criadas ainda a tabela `No` para armazenar as coordenadas dos nós provenientes do cálculo, a tabela `Furo` e a tabela `Perfil` para armazenar os perfis que são utilizados pelas barras.

5.3.1 Tabela `Torre`

A tabela `Torre` possui todos os elementos necessários para o desenho da torre metálica. A estrutura da tabela pode ser visualizada na Figura 5-3. A tabela `Torre` relaciona-se com as outras tabelas através dos códigos das mesmas e possui ainda dados específicos do objeto torre a ser desenhado.

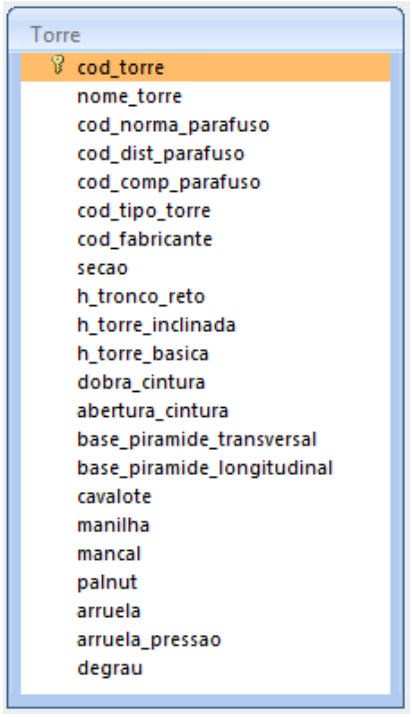


Figura 5-3 - Tabela `Torre`

O campo `seção` armazena a informação sobre o tipo de seção da torre. A torre pode ter seção quadrada, retangular ou triangular.

Os campos `h_tronco_reto` e `h_torre_inclinada` armazenam informações sobre a geometria básica da torre conforme a Figura 1-2 onde são denominados cabeça e tronco inferior respectivamente. Já o campo `h_torre_basica` armazena a distância da cabeça da torre até à primeira extensão, ou seja, o primeiro ponto da torre onde os pés podem ser encaixados, denominado corpo básico conforme mostrado na Figura 1-2.

Os campos `abertura_cintura`, `base_piramide_transversal` e `base_piramide_longitudinal` também armazenam informações básicas da geometria da torre. A abertura na cintura é a abertura onde ocorre a diferença de inclinação, na passagem da cabeça para o tronco inferior da torre. As bases piramidais são as aberturas entre os pés da torre montada com todas as extensões possíveis e com os maiores pés, ou seja, a torre mais alta da família de torres.

Todos os campos citados acima são preenchidos com dados provenientes do cálculo da estrutura.

Os dados do campo `dobra_cintura` são preenchidos pelo projetista da torre. É a distância entre o eixo de dobra da cintura e o eixo de cálculo estabelecido pelo cálculo que é representado pelo eixo principal de furação da barra horizontal da cintura. O detalhe de dobra da cintura é definido pelo projetista quando ele faz o estudo da torre em função dos perfis determinados pelo cálculo. A distância é destacada pelo círculo vermelho que pode ser visualizado na Figura 5-4.

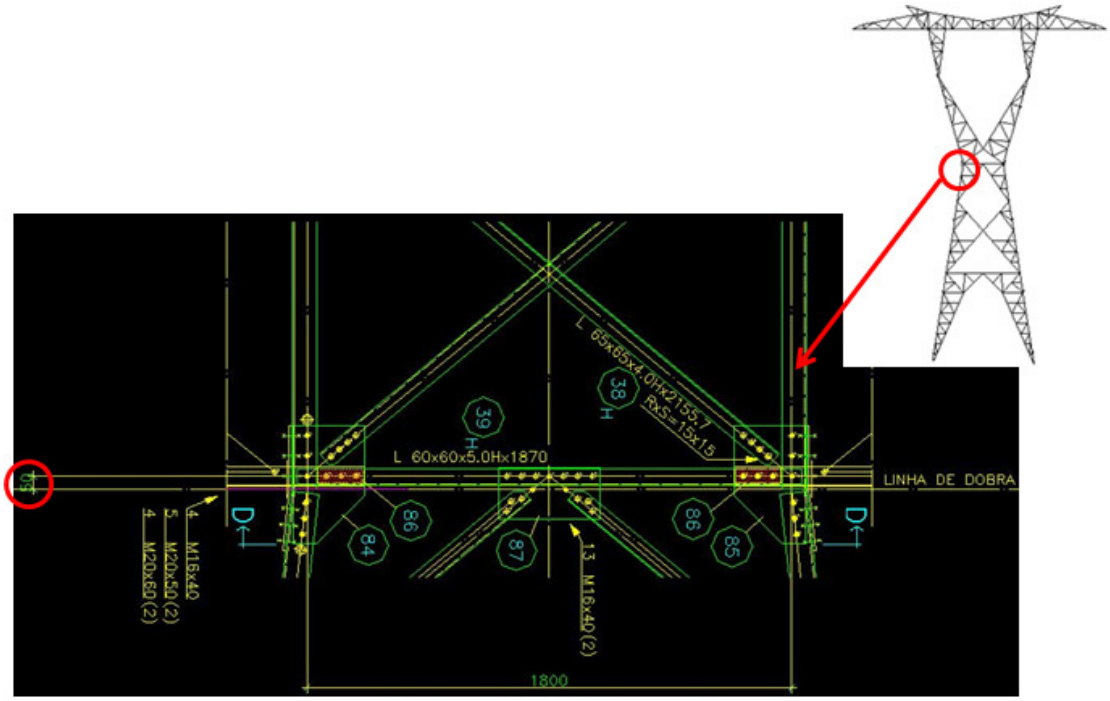


Figura 5-4 - Detalhe de dobra na cintura

Os demais campos são preenchidos a partir das especificações utilizadas por uma determinada torre. As especificações da torre, que são de responsabilidade do cliente, determinam se a torre irá conter e qual o tipo de cavalote, manilha, mancal, palnut, arruela, arruela de pressão e degrau.

O cavalote, a manilha e o mancal são peças utilizadas para fazer o ataque dos cabos condutores de energia e do pára-raios. O cliente normalmente decide como será o ataque em determinada torre e o projetista define qual peça utilizar em função da carga de ruptura. Quando o cliente decide utilizar o mancal é ainda necessário utilizar o cavalote ou a manilha.

A arruela protege o material, a arruela de pressão funciona como contraporca e o "*palnut*" é um tipo de contra porca. O cliente decide ainda se serão utilizados parafusos degraus, que servem para que uma pessoa possa subir na torre em caso de manutenção. Normalmente eles são colocados a partir de três metros de altura.

5.3.2 Tabela `Barra`

A tabela `Barra` pode ser considerada como a segunda tabela mais importante da estrutura de banco de dados. A estrutura da tabela é mostrada na Figura 5-5. A tabela `Barra` relaciona-se com as outras tabelas através dos códigos das mesmas e possui ainda dados específicos do objeto barra a ser desenhado.

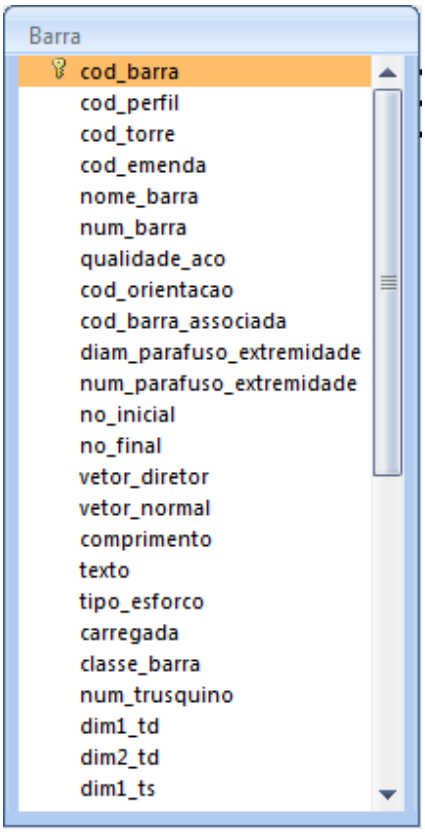


Figura 5-5 - Tabela `Barra`

Toda barra da estrutura contém um nome e um número para a sua identificação que ficam armazenados nos campos `nome_barra` e `num_barra`. Esses campos serão obtidos do cálculo estrutural para facilitar futuras verificações.

A qualidade do aço ficará armazenada no campo `qualidade_aco` e também será obtida do cálculo estrutural.

O campo `cod_orientação` armazena a informação de como a cantoneira será posicionada. Ela pode estar atrás da barra a ser conectada, à frente ou no mesmo plano no caso de a ligação ser realizada através de chapas.

O campo de `cod_barra_associada` é utilizado para o caso de que a cantoneira seja dupla, ou seja, no caso de o cálculo determinar que são necessários dois perfis para suportar o esforço.

Os `campos dim_parafuso_extremidade` e `num_parafuso_extremidade` armazenam respectivamente o diâmetro e o número de parafusos que são necessários nas extremidades da barra. Esses dados também são provenientes da etapa de dimensionamento do cálculo.

Os nós iniciais e finais das barras obtidos do dimensionamento servem no detalhamento para definir o primeiro parafuso da ligação que está no eixo de furação principal da barra. O comprimento armazenado é a distância entre os dois nós mais a distâncias dos nós até as bordas das cantoneiras. Essa distância depende do diâmetro do parafuso e da especificação do cliente. O tipo de esforço também pode interferir na distância uma vez que dependendo do tipo de esforço pode ser necessária uma segurança extra.

O campo `vetor_diretor` indica no espaço a direção da barra e o campo `vetor_normal` armazena a normal à aba pela qual a barra é ligada a outra.

O campo `carregada` armazena a informação se a barra é carregada ou descarregada. Barra carregada é uma barra que foi dimensionada para suportar algum esforço calculado na etapa de análise estrutural. A barra não carregada é a barra posicionada para evitar a flambagem local de determinada barra. Ela serve apenas de contraventamento.

Os campos de `distâncias` e que começam com o prefixo `dim` estão relacionados com as distâncias dos recortes que poderão ser feitos nas cantoneiras. A cantoneira pode ter um recorte simples quando apenas uma aba é recortada ou ainda o recorte duplo quando duas abas são recortadas. Os recortes duplos são indesejados pela dificuldade de manipulação na fábrica.

O número de trusquinos, que são os eixos de furação, também é armazenado na tabela `barra` assim como o valor de cada trusquino. O campo `num_trusquino` armazena a quantidade de trusquinos existente na barra. Já os campos `trusquino1`, `trusquino2` até o campo `trusquino8`, armazenam os valores de cada trusquino existente na barra. Foi considerado que oito trusquinos é uma quantidade razoável para quantidade máxima de trusquinos. O trusquino número um é o trusquino principal da barra, ou seja, aquela que contém os furos que estão relacionados com os nós do cálculo. Esse é o trusquino mais importante da barra.

5.3.3 Tabela `Emenda`

A tabela `Emenda` armazena informações sobre as emendas. A estrutura da tabela `Emenda` é visualizada na Figura 5-6.

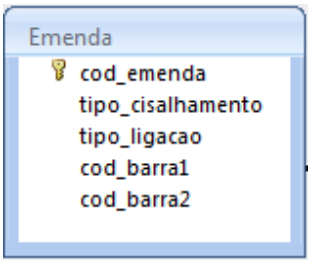


Figura 5-6 - Tabela `Emenda`

É importante ainda salientar a diferença existente entre emenda e ligação. A emenda aparece quando ocorre ligação entre uma mesma barra e a ligação ocorre quando duas ou mais barras são conectadas. A ligação entre uma mesma barra ocorre quando o tamanho da barra supera o tamanho máximo permitido ou quando o perfil da barra é diferente em um determinado trecho. O tamanho máximo permitido é definido em função de variáveis como, por exemplo, a viabilidade de transportar e trabalhar com a peça e é normalmente limitado a seis metros. O tamanho máximo permitido é informado pelo cliente.

O campo `tipo_ligacao` armazena informações sobre qual é o tipo de ligação da emenda. A emenda pode ser por sobreposição ou topo.

A emenda por sobreposição é normalmente utilizada quando existe uma grande diferença entre os perfis para manter os eixos de cálculo próximos. Neste caso uma cantoneira "entra" dentro da outra. A cantoneira que "entra" deve ser chanfrada para que as abas das duas cantoneiras fiquem "coladas".

A emenda de topo pode ser feita pelo alinhamento interno ou externo. O alinhamento externo é quando as cantoneiras são alinhadas pela parte de fora da aba e o alinhamento interno ocorre quando elas são alinhadas pela parte de dentro da espessura. O alinhamento interno é o mais utilizado quando o cisalhamento é duplo. Existe a necessidade de utilizar uma pequena chapa denominada calço quando existe uma diferença de espessura entre as cantoneiras maior do que 2 mm.

O campo `tipo_cisalhamento` armazena o tipo de cisalhamento da emenda. A emenda pode ser de cisalhamento duplo quando existem duas seções de cisalhamento ou de cisalhamento simples. Normalmente, no cisalhamento simples, a emenda contém uma cantoneira interna, também chamada de cobrejunto, com o mesmo perfil da barra que está sendo ligada. Já a emenda de cisalhamento duplo contém, além do cobrejunto, duas chapas localizadas nas abas da parte externa da cantoneira.

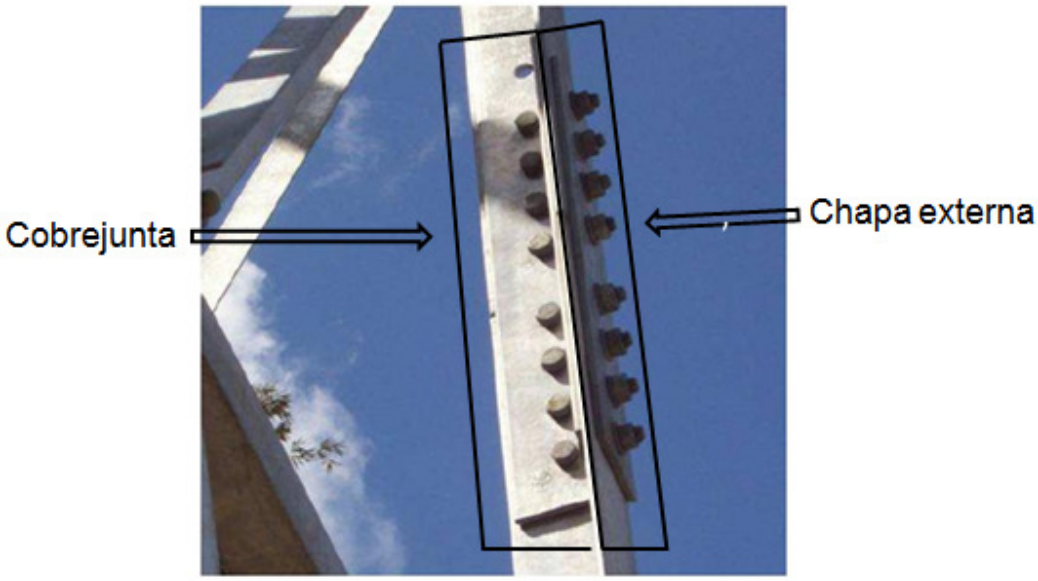


Figura 5-7 - Detalhe da emenda. Extraída de Silva, J.B.G.F & all (2009)

5.3.4 Tabela `Ligação`

A tabela `Ligação` armazena as informações que foram julgadas necessárias para o detalhamento das ligações conforme a Figura 5-8.

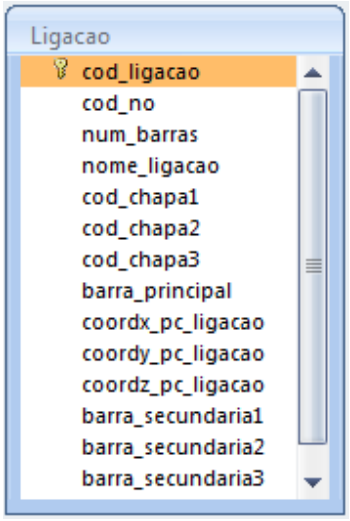


Figura 5-8 - Tabela `Ligação`

O campo `cod_no` armazena a informação de qual é o nó que está relacionado com a ligação. Foi considerado que toda ligação possui um nó de referência.

O campo `nome_ligacao` armazena o nome da ligação em questão.

O campo `num_barra` armazena o número de barras que chega à ligação, sendo o campo `barra_principal` responsável por armazenar a barra que é considerada a principal da ligação. Essa barra principal pode ser, por exemplo, o montante quando a ligação é feita nele. Os campos de `barra_secundaria` armazenam as outras barras que chegam à ligação. Uma ligação das treliças e barras horizontais com o montante é mostrada na Figura 5-9.

Figura 5-9- Ligação. Extraída de Silva, J.B.G.F & all (2009)

Os campos `coordx_pc_ligacao`, `coordy_pc_ligacao` e `coordz_pc_ligacao` armazenam as coordenadas do ponto de cálculo da ligação. Dependendo da face da ligação, as coordenadas do detalhamento da ligação são calculadas.

5.3.5 Tabela `Chapa`

As chapas são utilizadas nas ligações, quando não há espaço na área líquida das cantoneiras para colocar todos os parafusos, que foram dimensionados pelo cálculo. As chapas pertencem às ligações ou emendas.

A tabela `Chapa` contém as informações necessárias para a representação gráfica de uma chapa. A estrutura da tabela está na Figura 5-10.

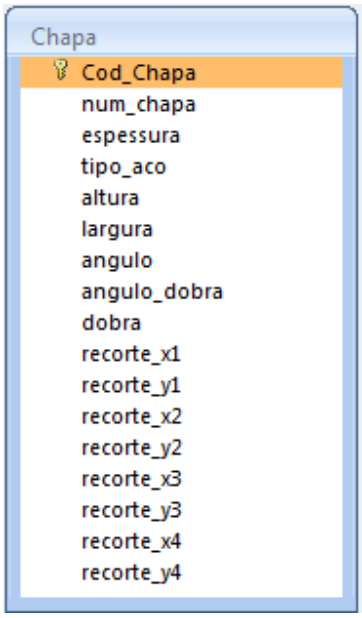


Figura 5-10 - Tabela `Chapa`

Além do código da chapa, estão armazenadas as informações número da chapa, espessura e tipo do aço utilizado.

O ângulo da chapa indica a inclinação da chapa com relação aos eixos x e y originais. O campo `dobra` armazena a informação que indica se a chapa possui uma dobra e o campo ângulo `dobra` armazena a informação que indica qual é o ângulo da dobra da chapa.

Os campos relacionados a recorte armazenam as informações dos recortes existentes em cada um dos cantos da chapa. A numeração indica qual canto da chapa será recortado e o x e o y indicam a direção que a chapa será recortada.

A Figura 5-11 mostra a representação gráfica de uma chapa utilizada para fazer a ligação das barras de contraventamento da vista de um pé da torre. Pode-se observar que a chapa representada pelo número 237 não possui recortes e por isso os campos relacionados a recorte estariam armazenados com o número zero. Em contrapartida, pode-se observar a representação gráfica das informações armazenadas nos campos `num_chapa`, `tipo_aco`, `altura`, `largura`, `ângulo`, `ângulo_dobra` e `dobra`.

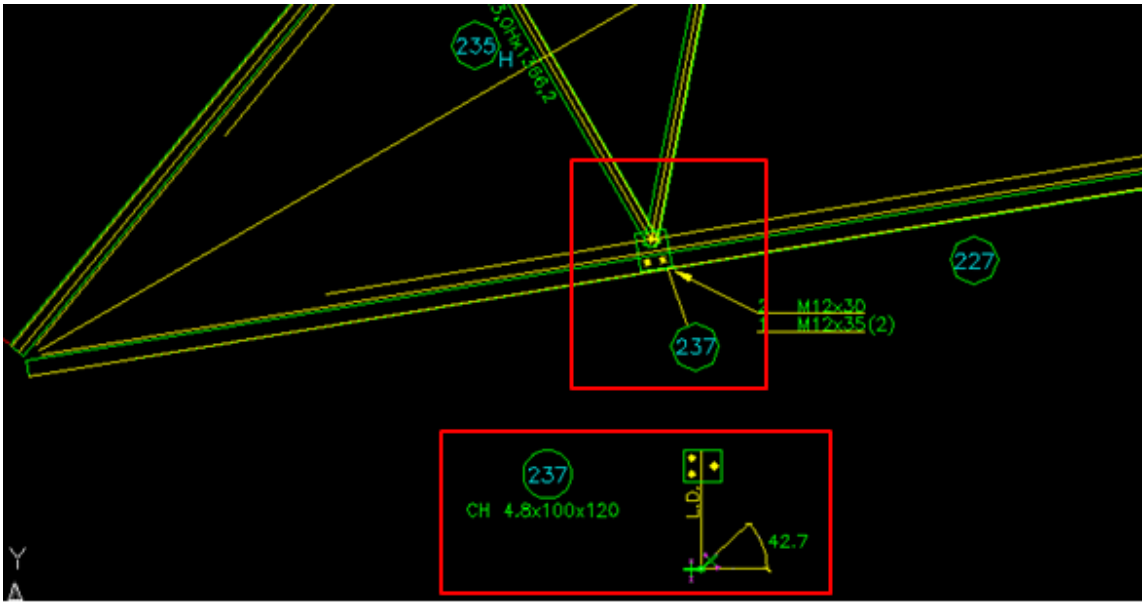


Figura 5-11 - Representação da Chapa

5.3.6 Tabela `Furo`

A tabela `Furo` armazena as informações necessárias para a representação gráfica de um furo no espaço. A estrutura da tabela é visualizada na Figura 5-12.

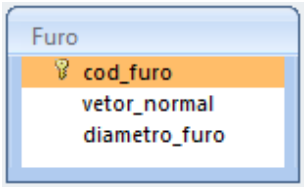


Figura 5-12 - Tabela `Furo`

Além do campo `diâmetro_furo`, que armazena o diâmetro do furo utilizado, foi criado o campo `vetor_normal`, que indica no espaço a localização do furo.

5.3.7 Tabela `Perfil`

A tabela `Perfil` contém todos os dados necessários para detalhar graficamente o perfil além das suas propriedades geométricas. A estrutura da tabela perfil está representada na Figura 5-13.

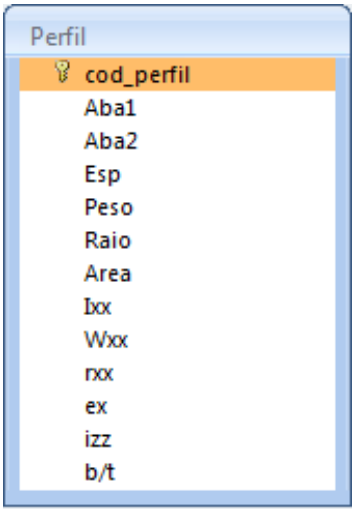


Figura 5-13 - Tabela `Perfil`

O campo `Ixx` refere-se ao momento de inércia em relação ao eixo x, o campo `Wxx` refere-se a módulo de resistência, o campo `rxx` refere-se ao raio de giração em relação ao eixo x, o campo `ex` refere-se à posição da borda externa da alma até o centro de gravidade da cantoneira, o campo `izz` refere-se ao raio de giração em relação ao eixo de menor inércia e o campo `b/t` refere-se a relação entre a parte plana da aba e a espessura da cantoneira.

Uma visualização de parte da tabela preenchida está na Figura 5-14

cod_perfil	Aba1	Aba2	Esp	Peso	Raio	Area	Ixx	Wxx	rxx	e
85	6350	6350	47625	457	63500	582	2277	496	198	
86	6350	6350	63500	610	63500	766	2930	646	195	
87	6350	6350	79375	740	63500	945	3530	790	193	
88	6350	6350	95250	880	63500	1120	4100	928	191	
89	6350	6350	127000	1150	63500	1450	5000	1200	188	
90	6500	5000	50000	435	60000	554	2320	514	205	
91	6500	5000	60000	516	60000	658	2720	610	203	
92	6500	5000	70000	596	60000	760	3110	703	202	
93	6500	5000	80000	675	60000	860	3480	793	201	
94	6500	6500	40000	402	90000	513	2010	421	198	
95	6500	6500	50000	495	90000	631	2470	521	198	
96	6500	6500	60000	591	90000	753	2920	621	197	
97	6500	6500	70000	683	90000	870	3340	718	196	
98	7000	7000	40000	436	90000	555	2540	491	214	
99	7000	7000	45000	485	90000	618	2850	555	215	
100	7000	7000	50000	537	90000	684	3130	611	214	
101	7000	7000	60000	638	90000	813	3690	727	213	
102	7000	7000	70000	738	90000	940	4230	841	212	
103	7000	7000	80000	836	90000	1060	4750	952	211	
104	7000	7000	90000	934	90000	1190	5260	1060	210	

Figura 5-14 - Tabela `Perfil` preenchida

5.3.8 Tabela `No`

A tabela `No` armazena as coordenadas dos nós e o código com o qual ele está relacionado. A estrutura da tabela encontra-se na Figura 5-15.

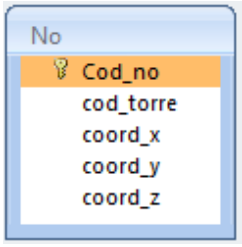


Figura 5-15 - Tabela `No`

5.3.9 Tabela `Tipo_Torre`

A tabela `Tipo_torre` é talvez a tabela mais simples do banco de dados. A estrutura da tabela é mostrada na Figura 5-16. Ela se relaciona com a tabela `Torre` através do campo `cod_tipo_torre`. O campo `nome_tipo_`torre armazena de que tipo é a torre. A torre pode ser, por exemplo, do tipo estaiada monomastro, tronco piramidal, delta, "*cross rope*" conforme mostrado na Figura 1-1.

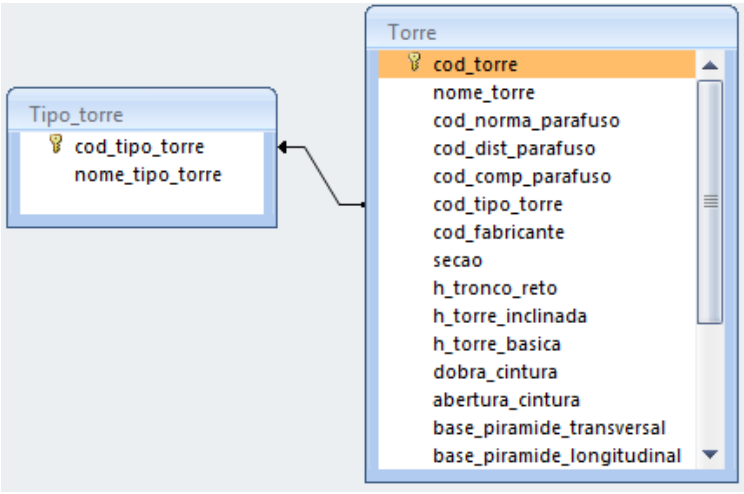


Figura 5-16 - Tabela `Tipo_Torre`

5.3.10 Tabelas sobre parafusos

As informações sobre parafusos não estão necessariamente conectadas. A especificação do cliente pode determinar, por exemplo, as mesmas distâncias mínimas, mas com diâmetros de parafusos diferentes ou comprimentos de rosca diferentes.

Diante do exposto, uma tabela para parafusos não seria suficiente. Foram então criadas três tabelas: as tabelas `Comprimento_parafuso`, `Distancia_parafuso` e `Norma_parafuso`. Cada uma dessas tabelas contém campos que estão relacionados entre si, mas que não estão necessariamente relacionados com os campos das outras tabelas.

A tabela `Distancia_parafuso` armazena as distâncias mínimas dos parafusos que serão adotadas pelo cliente. A estrutura da tabela está na Figura 5-17.

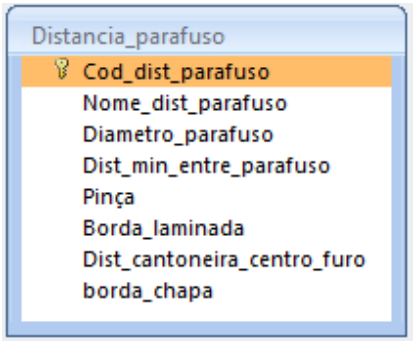


Figura 5-17 - Tabela `Distancia_parafuso`

O campo `Cod_dist_parafuso` faz a conexão da tabela `Cod_dist_parafuso` com a tabela torre, ou seja, informa quais serão as distâncias mínimas adotadas em determinada torre. As distâncias estão ainda relacionadas com o diâmetro do parafuso. Cada cliente deve informar as distâncias mínimas utilizadas para cada diâmetro de parafuso.

Os diâmetros de parafusos normalmente utilizados são:

- Parafuso M12 – 12 mm.
- Parafuso M16 – 16 mm.
- Parafuso M20 – 20 mm.
- Parafuso M22 – 22 mm.
- Parafuso M24 – 24 mm.

As distâncias mínimas estão ilustradas na Figura 5-18. Lembrando que o campo `Dist_min_entre_parafuso` está ilustrado como `Entre Parafusos`, o campo `Pinça` está ilustrado como `Borda Cortada` e o campo `Dist_cantoneira_centro_furo` está ilustrado como `Outra Cantoneira`.

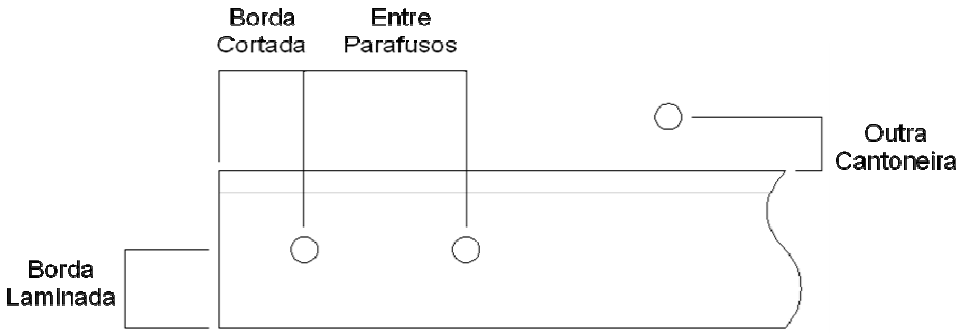


Figura 5-18 - Distâncias mínimas

O campo `borda_chapa` armazena a distância mínima que deverá ser utilizada do centro do parafuso até a borda da chapa.

A tabela `Comprimento_parafuso` armazena o comprimento da rosca e o passo do comprimento.

A tabela `Norma_parafuso` armazena informações sobre a cabeça do parafuso e da porca utilizada. A estrutura da tabela é mostrada na Figura 5-19.

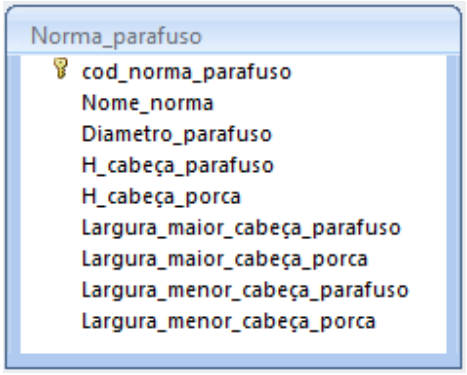


Figura 5-19 - Tabela `Norma_parafuso`

Assim como na tabela `Distancia_parafuso`, a tabela `Norma_parafuso` deve conter um registro para cada tipo de parafuso utilizado.

As informações armazenadas nessa tabela são importantes para definição do comprimento total do parafuso e do eixo de furação principal da peça. As alturas do parafuso e da porca fazem parte da definição do comprimento do parafuso. As larguras do parafuso e da porca ajudam a definir um eixo de furação que permite a montagem em campo sem dificuldades extras.

5.3.11 Tabelas sobre o desenho

Algumas tabelas foram criadas e relacionadas entre si para criar os formatos nos quais os desenhos de detalhamento da torre serão entregues aos clientes finais. Essas tabelas são `Desenho, Formato, Nota e Legenda`.

A tabela `Desenho` é a principal tabela entre elas e possui uma referência para cada uma das outras tabelas. As referências são feitas através dos códigos das tabelas conforme ilustrado na Figura 5-20.

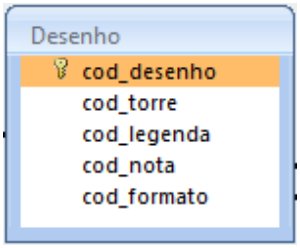


Figura 5-20 - Tabela `Desenho`

A tabela `Formato` possui as informações necessárias para a representação gráfica do formato escolhido.

Além das coordenadas de inserção, a tabela armazena informações sobre o tipo de formato que será utilizado, a altura, a largura e as margens. A estrutura da tabela é mostrada na Figura 5-21.

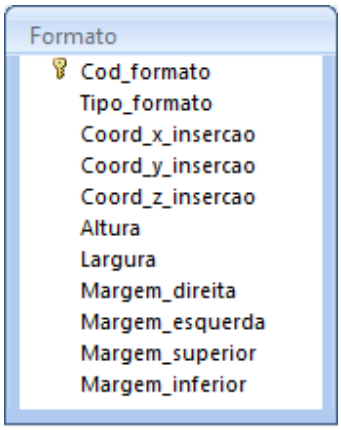


Figura 5-21 - Tabela `Formato`

A tabela `Nota` armazena o conteúdo e as coordenadas de inserção conforme a Figura 5-22.

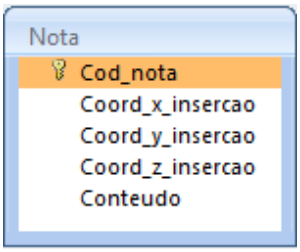


Figura 5-22- Tabela `Nota`

Os dados da tabela `Nota` armazenam informações sobre o desenho e as referências de onde encontrar mais informações. Um exemplo de notas de um desenho de detalhamento está na Figura 5-23.

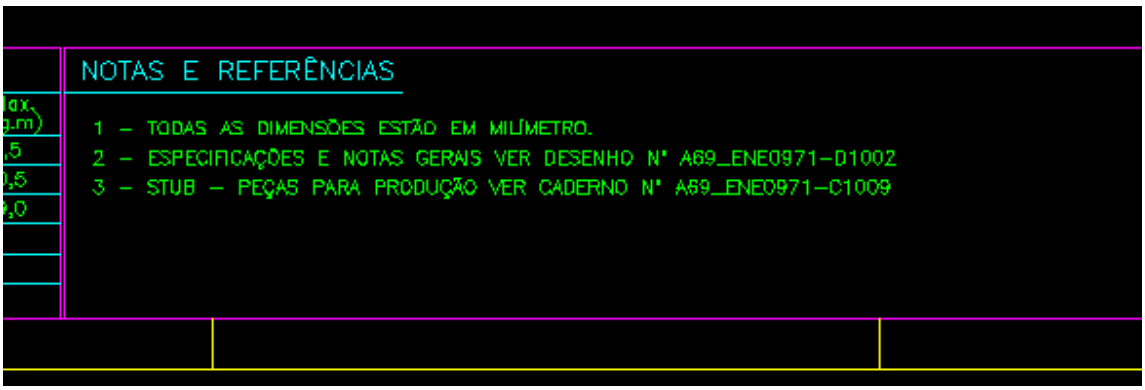


Figura 5-23 - Notas

A tabela `Legenda` armazena as informações restantes do desenho, tais como, o nome do projeto, o número da folha, o número do desenho, a escala utilizada, o engenheiro responsável pelo projeto e as datas de execução, verificação, aprovação, projeto e cálculo. A estrutura da tabela `Legenda` está ilustrada na Figura 5-24

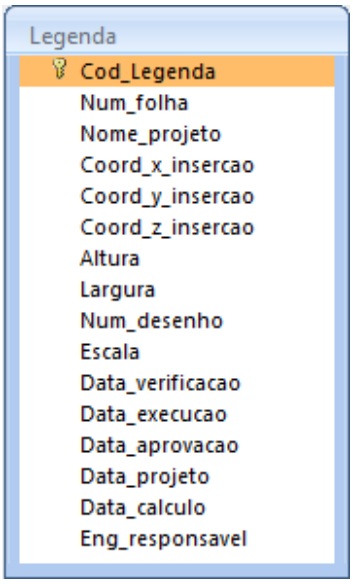


Figura 5-24 - Tabela `Legenda`

5.4 O Relacionamento entre as tabelas

Os relacionamentos entre tabelas foram criados com finalidade de fazer a ligação entre as informações que estão armazenadas em cada uma das tabelas da estrutura de banco de dados.

O principal tipo de relacionamento utilizado na criação da estrutura de banco de dados desse trabalho foi o relacionamento Um para Vários. O relacionamento Um para Vários é o relacionamento entre um campo que é chave primária e um campo que não é chave primária.

5.4.1 Relacionamentos da tabela `Torre`

Por ser a principal tabela da estrutura do banco de dados seria normal que a tabela `Torre` fosse a tabela com mais relacionamentos. É exatamente o que acontece. A tabela `Torre` se relaciona com a tabela `Barra`, `No`, `Desenho`, `Distancia_parafuso`, `Norma_parafuso`, `Comprimento_parafuso` e `Tipo_torre`.

O relacionamento das Tabelas com a tabela `Torre` é ilustrado na Figura 5-25.

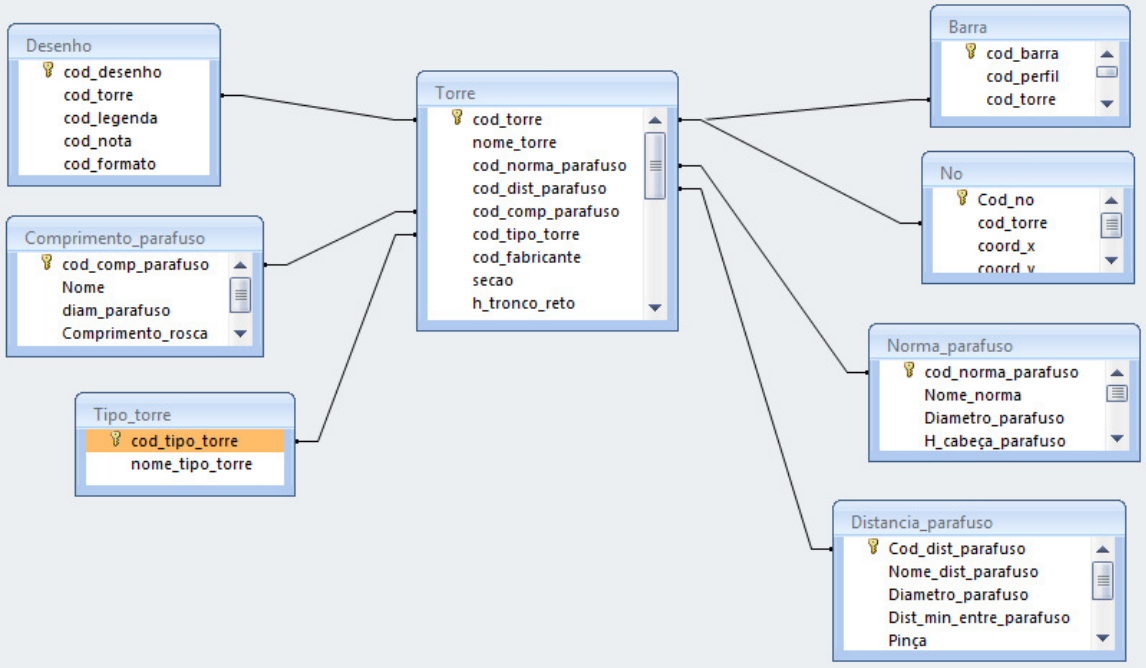


Figura 5-25 - Relacionamentos da tabela `Torre`

A torre a ser analisada deve seguir algumas normas e/ou especificações estabelecidas pelos clientes. Informações a respeito das especificações do cliente sobre parafusos que serão utilizados na torre estão armazenadas nas tabelas `Norma_parafuso`, `Comprimento_parafuso` e `Distancia_parafuso`. Considerando o fato de que cada uma dessas tabelas armazena uma quantidade finita de registros que pode aumentar à medida que novas especificações forem criadas e que várias torres podem utilizar as mesmas especificações, foi estabelecido que o relacionamento entre essas tabelas e a tabela `Torre` é o relacionamento Um para Vários. A chave primária de cada um dos relacionamentos está nas tabelas e a chave estrangeira, ou seja, o lado Vários do relacionamento está na tabela `Torre`.

O relacionamento Um para Vários também foi escolhido para a ligação entre a tabela `Torre` e `Tipo_Torre`. Nesse caso, o tipo de torre não será repetido, pois é chave primária, e várias torres poderão ter o mesmo tipo, chave estrangeira.

No caso das tabelas `Barra`, `No` e `Desenho`, também foi utilizado o relacionamento Um para Vários, mas dessa vez com a chave primária localizada na tabela `Torre`. Cada torre possui várias barras e uma barra pertence apenas a uma torre. O campo `cod_torre` é, então, único na tabela `Torre` e pode ter valores repetidos na tabela `Barra`. O mesmo acontece com os nós e desenhos da torre. Cada torre possui vários nós e um nó pertence apenas a uma torre. Cada torre possui vários desenhos e um desenho pertence apenas a uma torre.

5.4.2 Relacionamentos da tabela `Barra`

A tabela `Barra`, assim como a tabela `Torre`, possui relacionamento com muitas tabelas da estrutura de banco de dados. Os relacionamentos da tabela `Barra` com as outras tabelas estão ilustrados na Figura 5-26.

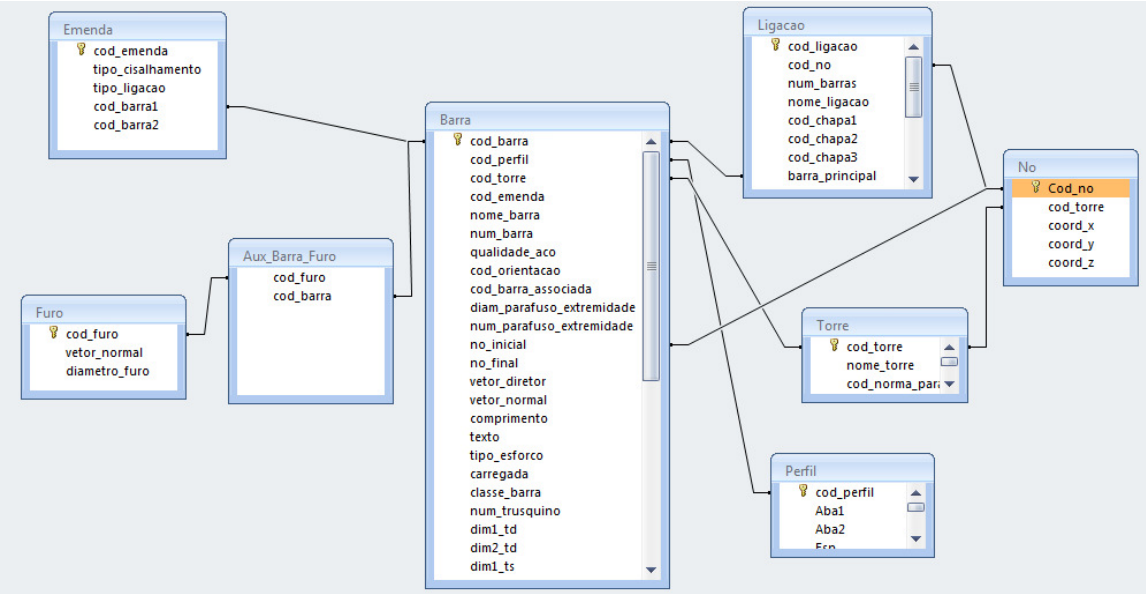


Figura 5-26 - Relacionamentos da tabela `Barra`

A tabela `Barra` se relaciona diretamente com a tabela `Emenda`, `Ligacao`, `Torre`, `Perfil` e `No`. Relaciona-se ainda indiretamente com a tabela `Furo`.

O relacionamento da tabela `Barra` com a tabela `Torre` já foi descrito no tópico anterior, relacionamento Um para Vários no qual a chave primária está localizada na tabela `Torre`. O relacionamento com a tabela `Perfil` é similar. Uma barra contém apenas um perfil e um perfil pode estar presente em várias barras diferentes. Com isso, o campo `cod_perfil`, que é o campo que relaciona as duas tabelas, é chave primária na tabela `Perfil` e chave estrangeira na tabela `Barra`. Os valores do campo `cod_perfil` podem ser repetidos na tabela `Barra`.

A tabela `No` possui um relacionamento similar às tabelas anteriores, mas ao invés de um relacionamento possui dois. Toda barra possui um nó inicial e um nó final e mais de uma barra pode ter o mesmo ponto inicial ou final. Foram criados então dois relacionamentos. Um relacionamento é do campo cod_no da tabela `No`, chave primária, com o campo `no_inicial` da tabela `Barra` e o outro é do mesmo campo `cod_no` da tabela `No`, também como chave primária, e o campo `no_final` da tabela `Barra`. Os dois relacionamentos criados são do tipo Um para Vários com a chave primária localizada na tabela `No`.

As tabelas `Emenda e Ligacao` possuem relacionamento do tipo Um para Vários com a tabela `Barra`, mas a chave primária desses relacionamentos está localizada na tabela `Barra`. No caso dessas tabelas, uma ligação possui várias barras e uma emenda também possui várias barras. A ligação é feita pelo campo `cod_barra` que é a chave primária da tabela `Barra`.

A tabela `Furo` possui um tipo de relacionamento com a tabela `Barra` diferente dos relacionamentos anteriores, o relacionamento Vários para Vários. Esse tipo de relacionamento acontece porque uma barra pode ter vários furos e um furo pode conter várias barras. Para transformar esse relacionamento em dois relacionamentos Um para Vários, foi criada a tabela auxiliar `Aux_Barra_Furo`. A tabela `Aux_Barra_Furo` contém dois campos auxiliares `cod_furo` e `cod_barra` que são utilizados para serem as chaves estrangeiras dos dois relacionamentos criados conforme pode ser visualizado na Figura 5-26.

5.4.3 Outros relacionamentos

Todos os desenhos possuem uma legenda, um formato e notas. A ligação é feita com o relacionamento um para vários. A tabela `Desenho` possui o relacionamento vários conforme mostrado na Figura 5-27, ou seja, na tabela `Desenho` podem existir vários registros de desenhos que possuem ligação com o mesmo registro das outras tabelas.

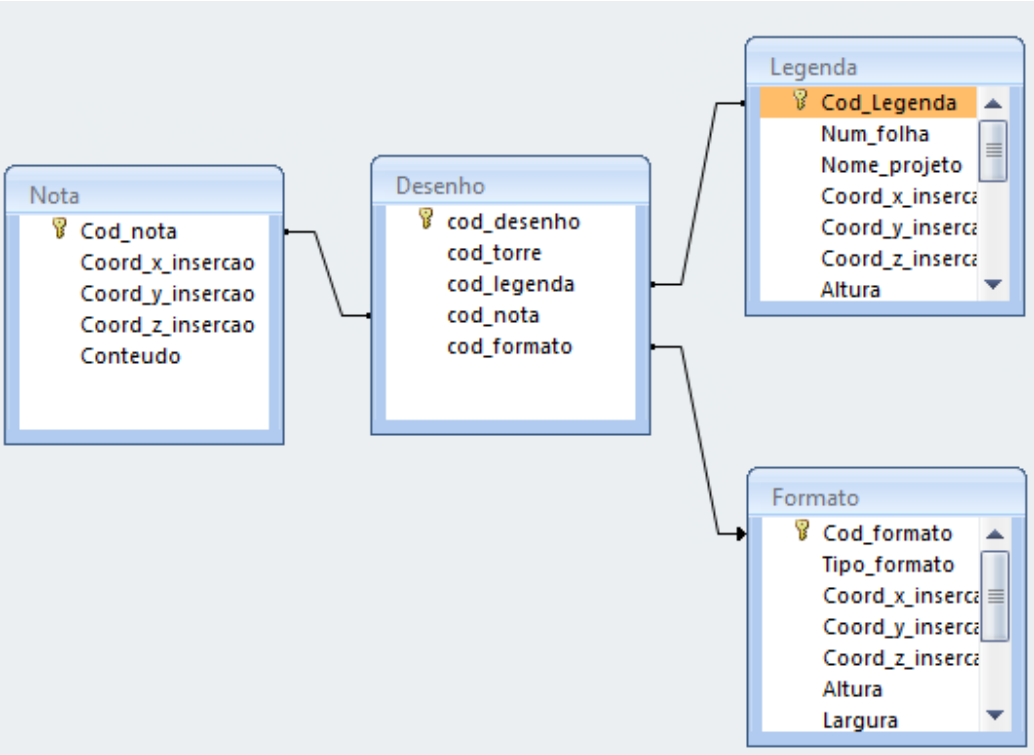


Figura 5-27 - Relacionamentos da tabela `Desenho`

Algumas ligações podem conter uma chapa. A tabela `Chapa` possui, portanto um relacionamento um para vários com a tabela `Ligacao`. A chave primária está localizada na tabela `Chapa`.

5.5 Povoamento do Banco de dados

O povoamento do banco de dados pode ser feito diretamente no sistema gerenciador de banco de dados, pelos projetistas através das interfaces desenvolvidas como, por exemplo, no comando `PParafuso` ou ainda através de arquivos de dados com configurações pré-definidas como no comando `AbrirArquivo`.

6. CONSIDERAÇÕES FINAIS

A automação de processos de fabricação é um anseio antigo das indústrias, com uma demanda especial na área de estruturas industrializadas, que ainda opera de uma forma quase artesanal. Ferramentas como o aplicativo desenvolvido para esta dissertação mostra uma nova perspectiva para a automação de tarefas e etapas do processo de fabricação de torres, ainda hoje executadas manualmente. Como resultado desta pesquisa espera-se uma significativa mudança em relação a qualidade do produto, melhoria da produtividade e da competitividade no setor.

Nesse capítulo são apresentadas algumas considerações sobre o trabalho desenvolvido, suas contribuições e suas limitações, e são feitas ainda propostas para o desenvolvimento de trabalhos futuros.

6.1 O Trabalho Desenvolvido e suas Contribuições

Neste trabalho são apresentadas as atividades realizadas no desenvolvimento de um sistema para a automação do detalhamento de torres metálicas. A etapa de estudo e formulação, aliada à concepção do escopo conceitual do sistema computacional foram de significativa

importância para a realização deste e de futuros trabalhos. A partir de estudos de sistemas CAD desenvolvidos anteriormente como em Malard (1998), Hutner (1998), Leite (2002), Leite (2007) e Magalhães (2002) foi construída uma estrutura independente de banco de dados, que associada a um aplicativo CAD permite automatizar o detalhamento de estruturas de torres metálicas.

O escopo conceitual do projeto do aplicativo TowerCAD prevê sua extensibilidade e aprimoramento através do desenvolvimento de novos algoritmos ou ainda da modificação e criação das novas tabelas na estrutura de banco de dados. Outros "problemas", fora da área de engenharia de estruturas, poderão usufruir dos avanços deste trabalho para automatizar processos utilizando a mesma tecnologia onde os dados estão representados em um banco de dados externo e a representação gráfica é feita em uma plataforma gráfica.

6.1.1 Contribuições Gerais

Uma importante contribuição deste trabalho foi a aplicação de um novo paradigma para o desenvolvimento de sistemas computacionais no padrão de arquitetura MVC, onde o modelo, a visualização e o controle são separados e funcionam independentemente. A inovação introduzida está na separação dos dados em um banco de dados externo com gerenciamento/controle independente da plataforma gráfica de gerenciamento/controle da representação gráfica. O controle do banco de dados, o controle do sistema CAD de visualização e do sistema de interfaces de comunicação entre eles opera também de forma programática independente. Esse novo paradigma permite que a estrutura de dados do sistema fique independente da plataforma gráfica e que uma possível migração de plataforma gráfica seja feita facilmente.

Esse trabalho é também pioneiro em uma nova linha de desenvolvimento tecnológico no PROPEEs, que inicia a construção de sistemas computacionais, utilizando a tecnologia .NET, através da API .NET para programação do AutoCAD. Essa mesma tecnologia .NET permite a interface de um banco de dados externo a uma plataforma gráfica através da tecnologia ADO.NET.

Outra importante contribuição da presente pesquisa foi a concepção de uma estrutura de dados independente da plataforma gráfica, baseada em tabelas bem definidas, capaz de atender a base de dados do aplicativo desenvolvido e ainda suportar novas implementações. Essas novas implementações podem ser desenvolvidas com a mesma estrutura da base de dados desenvolvida, utilizando a mesma estrutura para criação de novas tabelas e relacionamentos. Esta estrutura de dados pode ainda ser reaproveitada para o desenvolvimento de uma nova base de dados relacionada a outro tipo problema.

A independência obtida entre dados, visualização e controle permite a flexibilização na escolha da plataforma gráfica a ser utilizada, facilita a implementação das representações gráficas, uma vez que toda a manipulação de dados é realizada programaticamente no banco de dados externo e a aplicação apenas define a visualização dos elementos. O mesmo banco de dados pode servir de base para aplicações desenvolvidas em diferentes

plataformas gráficas. Permite também a migração mais fácil para versões mais recentes dos *softwares*, bastando uma atualização e recompilamento do código fonte. Esta independência traz ainda como vantagem a facilidade para *backups* e por isso se torna uma ferramenta mais eficaz.

6.1.2 Contribuições Específicas para a Automação de Processos Produtivos

A construção de um sistema global de automação foi iniciada pelo lançamento de estruturas de bases de dados capazes de suportar o desenvolvimento de sistemas integrados, contemplando desde a entrada de dados ou modelagem até a fabricação dos componentes da estrutura.

O imenso potencial desta tecnologia para automatizar cada vez mais o processo produtivo pode ser destacado. O aumento da produtividade, a precisão dos resultados e a melhoria da qualidade dos produtos são alguns dos resultados esperados.

A estrutura de banco de dados criada foi concebida para permitir novas implementações necessárias ao desenvolvimento completo de pré-processadores e pós-processadores capazes de fazer a integração entre os sistemas do processo produtivo. A estrutura de banco de dados é responsável pela manutenção, transmissão e pelo gerenciamento dos dados necessários às demais etapas deste processo.

O módulo Controle, que promove o gerenciamento dos dados, importação/exportação, leitura, edição, etc..., a partir de interfaces gráficas pode ser considerado uma grande contribuição tanto para o projeto global de pesquisa na área de torres quanto para futuros projetos de automação, que utilizarem plataformas CAD. A implantação da separação entre a plataforma gráfica e o armazenamento dos dados não seria possível sem a implementação do módulo Controle.

6.1.3 Contribuições Específicas para a Indústria de Torres Metálicas

O sistema desenvolvido construiu as bases para um trabalho mais amplo de detalhamento de estruturas de torres metálicas. A estrutura de banco de dados criada foi concebida prevendo implementações futuras, que permitam o detalhamento de outras partes da estrutura. Novos elementos e novas tabelas podem ainda ser criados, utilizando a estrutura atual como base.

O TowerCAD proporciona significativos aumentos de eficiência, produtividade e segurança aos projetos de detalhamento de torres metálicas. A utilização do detalhamento automático em substituição ao detalhamento manual traz importantes ganhos de produtividade e

qualidade aos projetos. As diferenças entre a estrutura detalhada e a estrutura real fabricada diminuem a níveis bastante aceitáveis.

Os comandos criados já estão sendo utilizados pelos projetistas da área Os primeiros impactos na produção de projetos de torres metálicas já podem ser observados pela redução do tempo gasto e maior precisão nos desenhos de detalhamento.

Foram criados módulos de interface com o projetista conforme proposto em Magalhães (2002). As interfaces amigáveis desenvolvidas dentro do AutoCAD permitem aos projetistas informar os parâmetros, que são necessários para o detalhamento de uma torre e que ainda não são calculados automaticamente. Os parâmetros informados são armazenados no banco de dados externo.

A interface CAM para equipamentos de controle numérico também é uma contribuição importante deste trabalho. O arquivo gerado contribui para automação da fabricação de torres metálicas. A interferência humana é minimizada e conseqüentemente a produtividade aumenta e a probabilidade de erros diminui.

Cabe observar que este trabalho foi o primeiro a produzir desenhos de detalhamento de torres metálicas para montagem. Em Magalhães (2002), somente haviam sido gerados desenhos de peças isoladas.

6.2 Limitações do Aplicativo e Propostas para Trabalhos Futuros

Nesta primeira etapa de desenvolvimento do sistema CAD denominado TowerCAD foram desenvolvidos comandos e a concepção para o detalhamento de uma torre metálica completa. No entanto, ainda existem alguns aspectos que necessitam de aprimoramento e outros que ainda devem ser implementados. As limitações do TowerCAD, considerando-se o desenvolvimento até aqui alcançado e as atividades a serem desenvolvidas em trabalhos futuros são descritas nos tópicos a seguir.

6.2.1 Detalhamento da estrutura

O detalhamento dos pés da torre foi escolhido como caso de estudo para esse trabalho e se encontra em fase avançada de desenvolvimento. Desenvolvimentos posteriormente permitirão o detalhamento de todos os tipos de geometria de pés e das outras parte da torre tais como tronco, cabeça, etc.... A geometria detalhada pode ser visualizada nas Figura 4-27 e 4.28..

O detalhamento das outras partes como, por exemplo, o tronco piramidal, as bases de extensões e o delta da torre ainda não foi inicializado, por outro lado a estrutura de banco de dados já foi elaborada para contemplá-las. Será necessário um estudo minucioso de

como detalhar as outras partes da torre assim como foi feito no caso dos pés. Outros comandos, assim como o comando `Cpe` citado na seção 4.4.5, devem ser criados à medida que o aplicativo for aperfeiçoado.

6.2.2 Ligações

O detalhamento de ligações, realizado nesta etapa de trabalho, restringiu-se às ligações dimensionadas com apenas um parafuso. Ligações dimensionadas com apenas um parafusos são comuns apenas na ligação de barras de contraventamento que são utilizadas normalmente nos pés da estrutura.

Ligações dimensionadas com mais de um parafuso em um ou mais trusquinos não foram detalhadas, mas serão necessárias para o detalhamento das outras partes da torre. As ligações com mais de um parafuso deverão ser parametrizados de acordo com a quantidade de parafusos e os perfis utilizados nas barras que serão ligadas. Deverá ser estudada a necessidade da utilização de chapas ou não para ligar as peças.

6.2.3 Emendas

As emendas entre as peças ainda não são detalhadas pelo aplicativo. Será necessário implementar um comando para detalhar as emendas com as distâncias mínimas utilizadas e possivelmente um ou dois trusquinos. As emendas também deverão ser parametrizadas de acordo com a quantidade de parafusos e o perfil.

6.2.4 Informações fornecidas pela projetista

Algumas informações solicitadas aos projetistas poderão ser calculadas automaticamente através da criação de algoritmos. Para isso devem ser estudados os dados necessários para o cálculo destas informações e, neste caso, alterar a estrutura de banco de dados criada para o aplicativo.

6.2.5 Interferência entre elementos

Os algoritmos para verificar a interferência entre os elementos ainda não foram implementados.

Será necessário elaborar algoritmos para verificar a existência de recortes e dobras para que as peças encaixem perfeitamente. Assim, os recortes e dobras poderão ser indicados nos desenhos e os desenhos de detalhamento serão produzidos cada vez mais rapidamente e com menor quantidade de erros.

6.2.6 Detalhamento 3D da estrutura

A estrutura de dados criada permite que desenvolvimentos futuros contemplem a visualização e o detalhamento 3D da estrutura. Os dados necessários para essa implementação se encontram disponíveis na estrutura. O detalhamento 3D da estrutura permitiria uma melhor visualização da estrutura a ser detalhada e uma curva de aprendizagem menor para novos projetistas.

6.2.7 Lista de Materiais

A lista de material ainda não é gerada pelo sistema TowerCAD. Com os dados armazenados no banco de dados externo, a lista de materiais pode ser gerada automaticamente com a implementação de um algoritmo.

7. REFERÊNCIAS BIBLIOGRÁFICAS

ABNT, 2008. NBR 8800 - Projeto de estruturas de aço e de estruturas mistas de aço e concreto de edifícios. Associação Brasileira de Normas Técnicas.

AISC, 2006. Manual of steel construction. 13th edition. American Institute of Steel Construction.

Autodesk, Inc., 2009. AutoCAD.NET Developer's Guide.

Autodesk, 2009a. Through the Interface: Autocad .NET. Disponível em http://through-the-interface.typepad.com/through_the_interface/autocad_net/ Acesso em Jun/2009.

Baldam, R., 2007. AutoCAD 2008 – Utilizando Totalmente.

Battisti, J., 2009. O Modelo Relacional de Dados. Disponível em <http://www.juliobattisti.com.br/artigos/office/modelorelacional_p1.asp> Acesso em Out/2009.

Gontijo, C.R., 1994. Contribuição à Análise e Projeto de Torres Autoportantes de Linhas de Transmissão. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Gontijo, H.D.O., 2009. Aplicação das tecnologias CAD, programação orientada a objetos e banco de dados na automação do processo de detalhamento de Torres Metálicas. Cilamce 2009.

Hutner, A., 1998. Modelador 3D para Estruturas Via CAD – Arquitetura do Sistema/Banco de Dados/Visualização. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Las Casas, R.S., 2008. Desenvolvimento de Tecnologias para EAD. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Leite, R.C.G., 2002. Automação do Processo de Modelamento e Detalhamento de Estruturas Pré-Moldadas de Concreto Armado Via Tecnologia CAD e Programação Orientada a Objeto. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Leite, R.C.G., 2007. Um Framework para Automação/Integração do Processo de Desenvolvimento de Projetos de Estruturas Reticuladas Tridimensionais. Belo Horizonte. (Tese de Doutorado – Escola de Engenharia da UFMG).

Macoratti, J.C., 2009. Mini-curso sobre ADO.NET. Disponível em <http://www.macoratti.net/ado_net1.htm> Acesso em Ago/2009

Magalhães, P.H.V, 2002. Automação do Processo de Modelamento e Detalhamento de Torres Metálicas Via Tecnologia CAD. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Malard, F.P., 1998. Modelador 3D para Estruturas Via CAD – Arquitetura do Sistema/Banco de Dados/Visualização. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Malard, F.P., 2009. ObjectARX & Dummies. Disponível em <http://arxdummies.blogspot.com/> Acesso em Jan/2009.

MCauley, C., 2000. Programming AutoCAD 2000® using ObjectARX™. Thomsom Learning™. 676p.

Microsoft, Inc., 2009. Microsoft .NET – Getting Started in .NET. Microsoft. Disponível em <http://www.microsoft.com/net/> Acesso em Fev/2009.

Microsoft, 2009a. Página principal MSDN. <http://msdn.microsoft.com/> Acesso em Fev/2009.

Sacramento, M.C., 2008. Introdução ao Oracle SQL. CADTEC/UFMG.

Sharp, J., 2008. Microsoft Visual C# 2008.

Silva, A.C., 2008. Automação de Projetos de Estruturas Metálicas em Plataforma CAD. Belo Horizonte. (Dissertação do mestrado – Escola de Engenharia da UFMG).

Silva, J.B.G.F & all, 2009. Comparison of general industry practices for lattice tower design and detailing. Electra n° 244, p 19-28.

Souza, R., 1995. Sistema de gestão da qualidade para empresas construtoras.

Tayor, A.G., 2003. SQL for Dummies 5th Edition.