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

"MODELADOR 3D PARA ESTRUTURAS DE BARRAS VIA CAD-
ARQUITETURA DO SISTEMA/BANCO DE DADOS/VISUALIZAÇÃO"

Fernando Poinho Malard

Dissertação apresentada ao Programa 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. José Ricardo Queiroz Franco
DEES/UFMG - (Orientador)

Prof. Armando César Campos Lavall
DEES/UFMG

Prof. Renato Cardoso Mesquita
UFMG

Profa. Jacqueline Maria Flor
DEES/UFMG

Engo. Roberval José Pimenta
CODEME ENGENHARIA LTDA

Belo Horizonte, 15 de julho de 1998
Aos membros do CADTEC e ao Departamento de Engenharia de Estruturas da Escola de Engenharia da Universidade Federal de Minas Gerais.
AGRADECIMENTOS

A Deus, à minha família e pessoas que amo, aos amigos, ao meu orientador, à equipe do CADTEC, aos professores da Escola de Engenharia e alunos do Curso de Mestrado em Engenharia de Estruturas.
RESUMO

Com o objetivo de ampliar a aplicação da Computação Gráfica na Engenharia de Estruturas, esta Dissertação de Mestrado apresenta o projeto e o desenvolvimento de um sistema CAD para modelamento de estruturas unifilares baseadas em Nós, Barras e Superfícies. Este sistema, implementado através dos conceitos de programação orientada a objetos e da linguagem C++, utiliza a plataforma gráfica do AutoCAD como base. O presente trabalho aplica os conceitos de entidades, objetos, banco de dados e eventos dentro do AutoCAD utilizando a biblioteca ObjectARX. Como parte adicional deste trabalho, são apresentados e aplicados conceitos de criação e manutenção da arquitetura e estrutura de aplicativos desenvolvidos em grupo. O controle e gerenciamento do aplicativo é apresentado de forma sistemática, organizada em módulos, utilizando bibliotecas de vínculo dinâmico (DLL's). São apresentados ainda conceitos de utilização de interfaces gráficas para tornar a comunicação com o usuário mais amigável e segura. Concluindo, são apresentados possíveis desdobramentos e implementações futuras deste e de outros aplicativos.
ABSTRACT

Aiming to broaden the application of the Computer Graphics on Structural Engineering design, the present M.Sc. Dissertation presents the project and the development of a CAD system for wired frame modeling of structures. The system is based upon the concepts of Nodes, Bars and Surfaces, which were implemented using object oriented programming and the programming language C++. AutoCAD is used as the graphical platform, allowing the application of the concepts of entities, objects, database and events through the ObjectARX library. Control and managing of the resulting CAD system is presented in a systemized form and organized in modules using Dynamic Linked Libraries (DLL’s). Concepts of graphical interfaces, which make communication with the user safer and more friendly are also presented. Finally, possible new extensions and future implementations for this and for other applications are proposed.
## SUMÁRIO

1. **INTRODUÇÃO**
   - 1.1. Engenharia Estrutural ........................................ 2
   - 1.2. Computação Gráfica ............................................. 4
   - 1.3. Modelamento Estrutural ......................................... 5
   - 1.4. CAD e a Engenharia ............................................. 6
   - 1.5. Convenções e Termos Técnicos ................................ 8
   - 1.6. Resumo do Conteúdo por Capítulo ............................. 10
   - 1.7. Objetivos .......................................................... 11

2. **ESTRUTURAÇÃO DOS RECURSOS PARA O PROJETO** ............... 14
   - 2.1. Mercado e Tendências Tecnológicas .......................... 15
   - 2.2. Metodologia de Projetos de Engenharia Estrutural .......... 16
   - 2.3. Otimização Utilizando a Computação Gráfica ............... 18
   - 2.4. Ferramentas da Computação Gráfica ............................ 20
   - 2.5. Plataformas: AutoCAD R14 e Windows NT 4.0 ............... 21
   - 2.6. Interfaces Visuais ............................................. 23
   - 2.7. Programação Orientada a Objetos (POO) ...................... 24
   - 2.8. Visual C++ 5.0 e ObjectARX 2.0 .............................. 27
   - 2.9. CADTEC ............................................................ 30
   - 2.10. CODEME Engenharia Ltda. .................................... 31

3. **FLUXOGRAMA DO PROJETO** .................................... 32
   - 3.1. Escopo Conceitual .............................................. 33
   - 3.2. Desenvolvimento Paralelo ..................................... 35
   - 3.3. Fluxograma do Projeto ........................................ 37

4. **DESENVOLVIMENTO DO PROJETO** ................................ 41
   - 4.1. Estruturação Inicial .......................................... 42
     - 4.1.1. Estrutura Global do Programa ............................ 43
     - 4.1.2. Reutilização de Rotinas ................................ 44
     - 4.1.3. Implementações Iniciais .................................. 45
     - 4.1.4. Testes e Monitoração de Rotinas Isoladas ............. 47
     - 4.1.5. Entidades Nativas e Entidades “Costumizadas” ....... 49
     - 4.1.6. Entidades, Objetos e o Banco de Dados do AutoCAD .. 51
   - 4.2. Criação de Entidades Básicas ............................... 56
     - 4.2.1. Entidade Node (Nó) ...................................... 57
       - 4.2.1.1. Variáveis Membros de Node .......................... 58
       - 4.2.1.2. Funções Membros de Node ............................ 58
       - 4.2.1.3. Representação Gráfica da Entidade Node .......... 60
       - 4.2.1.4. Lista de Barras ..................................... 62
       - 4.2.1.5. Verificações de Consistência ...................... 64
       - 4.2.1.6. Restrição Nodal ...................................... 65
4.2.2. Entidade Bar (Barra) .................................................. 68
  4.2.2.1. Variáveis Membros de Bar .................................. 69
  4.2.2.2. Funções Membros de Bar .................................. 70
  4.2.2.3. Representação Gráfica da Entidade Bar .................. 73
  4.2.2.4. Identificação de Nós e de Pontos Final e Inicial ....... 75
  4.2.2.5. Verificações de Consistência ............................ 76
4.2.3. Comportamento Conjunto de Nó e Barra ....................... 78
4.2.4. Entidade Face (Superfície) ................................. 82
  4.2.4.1. Variáveis Membros de Face ............................... 84
  4.2.4.2. Funções Membros de Face ................................ 85
  4.2.4.3. Representação Gráfica da Entidade Face ................. 87
  4.2.4.4. Verificações de Consistência ............................ 87
  4.2.4.5. Criação Individual de Superfícies ....................... 90
  4.2.4.6. Criação Automática de Superfícies ....................... 91

4.3. Implementações Avançadas ........................................ 92
  4.3.1. Operações no Banco de Dados do AutoCAD .................. 93
  4.3.2. Numeração Automática ........................................ 94
    4.3.2.1. Definição do Objeto Config1 ............................ 95
    4.3.2.2. Criação do Objeto Config1 ............................... 96
    4.3.2.3. Implementação da Numeração Automática nas Entidades ......................................................................... 97
  4.3.3. Retores de Entidades ............................................. 98
  4.3.4. Retores de Comandos ............................................. 101
  4.3.5. Edição de Entidades .............................................. 102
  4.3.6. Cópia de Entidades ............................................... 106
    4.3.6.1. Numeração Após a Cópia .................................. 107
    4.3.6.2. Ajuste das Entidades Após a Cópia ......................... 108
  4.3.7. O Comando OFFSET ............................................... 113
  4.3.8. Renumeração ...................................................... 115

4.4. Carregamentos ...................................................... 120
  4.4.1. Casos de Carregamento .......................................... 121
  4.4.2. O Objeto Casos .................................................. 122
    4.4.2.1. Definição do Objeto Casos .................................. 123
    4.4.2.2. Criação do Objeto Casos .................................. 124
    4.4.2.3. Manipulação do Objeto Casos ............................... 126
  4.4.3. O Objeto CurrCase .............................................. 131
    4.4.3.1. Definição do Objeto CurrCase ............................ 131
    4.4.3.2. Criação do Objeto CurrCase ............................... 133
    4.4.3.3. Manipulação do Objeto CurrCase ............................ 134
  4.4.4. A Função removeCaso ............................................. 136
  4.4.5. O Comando CASOS ............................................... 137
  4.4.6. Carga Concentrada na Entidade Node ....................... 142
    4.4.6.1. Variáveis e Funções ...................................... 143
    4.4.6.2. O Comando CLNODE ......................................... 147
    4.4.6.3. Cópia de Cargas Concentradas ............................ 149
  4.4.7. Carga Distribuída na Entidade Face ......................... 150
    4.4.7.1. Variáveis e Funções ...................................... 151
    4.4.7.2. O Comando CLFACE ......................................... 152
4.4.7.3. Cópia de Cargas Distribuídas .............................................. 154
4.5. Controle de Visualização .......................................................... 155
  4.5.1. Implementação do Controle nas Entidades ................................ 157
  4.5.2. O Comando DISPLAY ......................................................... 159
4.6. Exportação de Dados .................................................................. 162
  4.6.1. Coleta de Dados das Entidades ............................................. 162
  4.6.2. Arquivo de Exportação ......................................................... 164
  4.6.3. O Comando EXDATA .......................................................... 167
4.7. Dispositivos Globais .................................................................. 169
  4.7.1. Módulo Central do Modelador ............................................... 170
  4.7.2. Uso de Barras de Ferramentas .............................................. 174

5. MODELADOR ESTRUTURAL ......................................................... 177
  5.1. Descrição dos Módulos ............................................................ 178
  5.2. Limitações ............................................................................. 181
  5.3. Utilização ............................................................................. 183

6. CONCLUSÕES E PROPOSTAS PARA TRABALHOS FUTUROS ............ 185
  6.1. Conclusões ............................................................................. 186
    6.1.1. Coerência Entre o Modelador e a Engenharia ....................... 186
    6.1.2. Eficiência, Produtividade e Segurança .................................. 187
    6.1.3. Impactos da Computação Gráfica ....................................... 189
  6.2. Propostas .............................................................................. 190
    6.2.1. Criação de Barras Intermediárias ....................................... 190
    6.2.2. Opções de Escala e Representação Gráfica ......................... 191
    6.2.3. Interfaces com Programas de Análise ................................. 192
    6.2.4. Rotinas de Detalhamento ................................................. 192

7. REFERÊNCIAS BIBLIOGRÁFICAS ............................................... 195
  7.1. Impressa ............................................................................... 196
  7.2. Internet ............................................................................. 197
  7.3. On-Line ............................................................................. 198

8. APÊNDICES .............................................................................. 199
  8.1. Tipos do ObjectARX ............................................................... 200
  8.2. Tutoriais ............................................................................. 201
    8.2.1. Grelha ........................................................................ 201
    8.2.2. Trelíça Plana ................................................................. 203
    8.2.3. Pórtico Espacial ............................................................. 205
  8.3. Versão HTML da Dissertação ................................................. 208
  8.4. Apresentação Multimídia ......................................................... 208
Capítulo 1

INTRODUÇÃO

Engenharia Estrutural
Computação Gráfica
Modelamento Estrutural
CAD e a Engenharia
Convenções e Termos Técnicos
Resumo do Conteúdo por Capítulo
Objetivos
1.1 - Engenharia Estrutural

Desde o aparecimento das primeiras técnicas para cálculo de estruturas, sejam elas de concreto, aço, madeira ou outros materiais alternativos, a evolução tecnológica sempre andou ao lado da engenharia fornecendo ferramentas e facilidades cada vez mais avançadas e eficientes.

Com o advento dos computadores, a engenharia experimentou um grande avanço no desenvolvimento de métodos de cálculo e projetos. O aparecimento de novas ferramentas de trabalho causou um grande impacto na produtividade e na precisão dos resultados obtidos.

As novas técnicas de engenharia, aliadas aos novos equipamentos, criaram condições para que os engenheiros pudesssem aprofundar seus estudos nos problemas de engenharia sem grandes preocupações com o tempo e custo de processamento, obtendo resultados mais precisos.

Dentro do processo evolutivo computacional, o advento dos microcomputadores pessoais causou uma revolução tecnológica sem precedentes. O impacto desta nova ferramenta no desenvolvimento de tecnologias ocorreu devido à fácil acessibilidade, operação relativamente simplificada em relação aos grandes computadores, à sua portabilidade e manutenção barata. Como consequência, aplicativos específicos para todas as áreas profissionais, inclusive para a engenharia estrutural, começaram a ser desenvolvidos.

Ganhou-se muito em desempenho e eficiência no cálculo estrutural e ocorreu um salto qualitativo no desenvolvimento de projetos e na forma de calcular estruturas de uma maneira mais prática, fácil e otimizada. O perfil do engenheiro começou também a mudar com o surgimento de novos requisitos além dos conhecimentos de engenharia. A aplicação da computação tornou-se essencial e hoje só pode ser feita por engenheiros capacitados em
programação ou então por equipes mescladas de engenheiros e técnicos computacionais.

Nos dias de hoje os programas de elementos finitos são capazes de realizar análises detalhadas em estruturas de grande complexidade. Os recursos disponíveis em tais programas vão desde processadores gráficos para concepção e definição da geometria até análise não linear de estruturas. Este nível de complexidade requer do novo engenheiro uma formação sólida em engenharia além de conhecimento de programação, computação gráfica e do próprio programa de análise a ser utilizado. Estes programas não podem ser tratados como base única de conhecimento e sim como ferramentas de apoio aos engenheiros buscando apenas o aprimoramento das técnicas de engenharia e a viabilização de cálculos mais avançados no mesmo espaço de tempo.

Enquanto as universidades preparam-se para disponibilizar uma formação compatível com estas novas tecnologias de “software”, críticas e questionamentos surgem sobre a aplicabilidade de programas tão complexos aos problemas mais simples de engenharia. Questiona-se ainda as dificuldades de se extrair informações sobre o comportamento de estruturas de programas genéricos como é o caso de muitos disponíveis no mercado onde, muitas vezes, a complexidade da entrada de dados inibe a utilização de tais programas para fins mais simplificados. Há mais ou menos duas décadas, alguns grupos de pesquisadores têm defendido o desenvolvimento de programas voltados para problemas específicos ou para problemas de engenharia mais triviais.

O presente trabalho, dentro desta filosofia, apresenta um sistema de modelamento 3D para estruturas de barras. O presente modelador será utilizado como um pré-processador da geometria unifilar da estrutura para posterior análise. A seguir, são apresentados alguns conceitos.
1.2 - Computação Gráfica

Do desenvolvimento das tecnologias computacionais surgiu a computação gráfica como forma de visualização, criação e manipulação de objetos virtuais. Neste novo contexto, começaram a aparecer dispositivos e aplicativos para tratamento destes objetos gráficos com o objetivo de estreitar o espaço entre a abstração da realidade nos computadores e a compreensão da realidade pelo ser humano.

Se inicialmente, a representação bidimensional era satisfatória, logo a necessidade de modelamentos gráficos mais representativos do objeto real forçou uma evolução da computação gráfica para o modelamento em três dimensões. Em alguns casos, a realidade física do objeto tridimensional pode ser simplificada em planos bidimensionais com o objetivo de simplificação.

O desenvolvimento e a rápida expansão da utilização da computação gráfica provocaram um avanço rápido na tecnologia dos computadores e nas técnicas computacionais. Processamentos gráficos são extremamente complexos e exigem grande desempenho do processador e uma capacidade razoável de memória dos computadores. Estes requisitos acabaram por contribuir com a aceleração do processo evolutivo das tecnologias computacionais.

O número de aplicativos lançados no mercado foi considerável, com ofertas de processamentos gráficos dos mais variados. Muitos dos pacotes gráficos lançados no mercado são usados como suporte para pré e pós-processamentos de programas já consagrados na área de análise estrutural.
1.3 - Modelamento Estrutural

Em geral, os métodos numéricos para cálculo estrutural e as técnicas computacionais recentes baseiam-se na discretização da estrutura em elementos finitos. Estes elementos podem ser tão simples como um ponto ou tão complexos como um elemento sólido. Ao analisar algumas técnicas, pode-se observar que, direta ou indiretamente, sempre se está lidando com objetos localizados no espaço ou num plano arbitrário. Mais especificamente, sempre se está tratando com coordenadas tridimensionais, planares ou lineares.

O modelamento estrutural é uma abstração da complexidade de uma estrutura para representar o seu comportamento físico. Um bom modelador deve ser capaz de representar, de uma forma sistemática e organizada, as diversas formas de estruturas. Tal representação ocorre a partir de uma tradução da estrutura real ou idealizada para parâmetros que possam ser interpretados no papel ou, mais recentemente no computador, para as devidas aplicações.

Com a tecnologia de computação gráfica disponível atualmente o pré-processamento numérico cedeu lugar ao modelamento gráfico que, basicamente, trata de um problema de representação gráfica da geometria da estrutura. Para problemas estruturais, o conjunto de informações sobre a geometria da estrutura é a parte mais significativa dos dados a serem armazenados. Os dados não geométricos podem ser armazenados ou não juntamente com os dados geométricos formando assim uma representação do modelo estrutural.

Entre várias vantagens, as mais imediatas do modelamento gráfico são a facilidade de reprodução do modelo estrutural, o aumento significativo de produtividade e a minimização de erros. Além destas, outras vantagens são aquelas resultantes de subprodutos deste modelamento gráfico, ou seja, o
modelamento gráfico tridimensional pode gerar inúmeros subprodutos de forma simplificada mantendo a coerência com os dados do modelamento completo. A complexidade do modelamento tridimensional passa a ser pouco significativa quando se leva em conta os ganhos de produtividade, segurança e confiabilidade dos produtos gerados a partir do modelamento em 3 dimensões. Além da facilidade do modelamento, o reaproveitamento de estruturas é imediato, bastando para isso que se tenha uma base de dados reutilizável e de edição habilitada.

1.4 - CAD e a Engenharia

Com o desenvolvimento da computação gráfica, novos sistemas computacionais apareceram para contribuir com o processo de modelamento e de projetos de engenharia e arquitetura, denominados tecnologia CAD (Computer Aided Design, ou seja, Desenho Assistido por Computador).

O CAD reúne um conjunto de ferramentas computacionais para a execução de tarefas de visualização, modelamento, desenho e detalhamento de projetos de uma maneira geral. Um bom aplicativo baseado no sistema CAD deve tornar a representação gráfica de uma estrutura o mais próximo possível da realidade.

Dentro do contexto de atividades interrelacionadas de um projeto, o CAD passou a exercer o importante papel de automatização de procedimentos, desde a concepção até a interface com o processo de manufatura utilizando o CAM (Computer Aided Manufacturing ou Manufatura Assistida por Computador).

As informações referentes à definição gráfica, carregamentos, textos e outras, podem ser armazenadas em bancos de dados para diferentes utilizações futuras. Estes bancos de dados podem ainda ser compartilhados
entre grupos de trabalhos mantendo, dessa forma, as informações referentes ao projeto global sempre atualizadas. Os sistemas CAD hoje permitem uma integração entre os programas de análise estrutural, o pré-processamento gráfico da estrutura e o pós-processamento dos resultados da análise.

A utilização do CAD no pré-processamento inclui o modelamento gráfico da estrutura, a edição e mudança da geometria, a verificação visual da consistência dos dados e eliminação de erros. Para a análise, o usuário pode selecionar o conjunto de dados armazenados em banco de dados e promover sua exportação no formato de leitura desejado.

Figura 1.4-1 – Integração CAD/CAM.
O pós-processamento dos resultados de análise através do CAD permite uma representação visual da distribuição de erros da solução, do modelamento de deformadas de estruturas, de distribuição de tensões e talvez, o mais importante, a associação de perfis e ligações em 3D e o detalhamento da estrutura.

Os recursos dos sistemas CAD, integrados com o CAM, viabilizam ainda a automação de todo o processo, desde a definição gráfica da estrutura até sua produção, formando um processo automatizado de produção (Figura 1.4-1).

1.5 - Convenções e Termos Técnicos

Como se trata de um projeto pioneiro neste ambiente de engenharia, é importante apresentar ao leitor as convenções e termos técnicos e científicos utilizados nesta Dissertação. Alguns destes termos são específicos de alguns aplicativos gráficos, programas de linguagens e palavras específicas de bibliotecas gráficas e linguagens de programação (Figura 1.5-1).

<p>| Programação Orientada a Objetos (POO) | Nova filosofia de programação que se utiliza de uma base de dados organizada em objetos, propriedades e funções. É possível herdar características e o reaproveitamento de código é bastante eficiente. |
| CAD (Computer Aided Design) | Desenho Assistido por Computador. |
| Rotinas | Grupo ou sequências de comandos. |
| Bibliotecas | Em programação, são conjuntos de rotinas agrupadas de forma a fornecer uma base de programação. |</p>
<table>
<thead>
<tr>
<th><strong>C++</strong></th>
<th>Linguagem de programação que permite a aplicação dos conceitos da programação orientada a objetos.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ObjectARX</strong></td>
<td>Biblioteca para programação dentro do AutoCAD através dos conceitos de POO e da linguagem C++.</td>
</tr>
<tr>
<td><strong>AutoCAD Release 14 (R14)</strong></td>
<td>Aplicativo CAD para projetos que permite a programação e costumização utilizando bibliotecas ObjectARX.</td>
</tr>
<tr>
<td><strong>Visual C++</strong></td>
<td>Compilador da Microsoft que permite compilar e gerar aplicativos utilizando a linguagem C++.</td>
</tr>
<tr>
<td><strong>Entidades</strong></td>
<td>Objetos gráficos do AutoCAD tais como: linha, círculo, texto.</td>
</tr>
<tr>
<td><strong>Layer</strong></td>
<td>Característica que pode ser atribuída a cada entidade para que estas possam ser separadas em diferentes camadas de desenho.</td>
</tr>
<tr>
<td><strong>Quadro de Diálogo (Dialog Box)</strong></td>
<td>Dispositivo visual e interativo que permite a comunicação entre o usuário e o programa.</td>
</tr>
<tr>
<td><strong>Barras de Ferramentas (ToolBars)</strong></td>
<td>Conjunto de botões que possibilitam um acesso mais rápido aos comandos de um aplicativo.</td>
</tr>
<tr>
<td><strong>Autodesk</strong></td>
<td>Fabricante do AutoCAD e da biblioteca de desenvolvimento ObjectARX.</td>
</tr>
<tr>
<td><strong>Windows NT 4.0</strong></td>
<td>Sistema operacional desenvolvido pela Microsoft.</td>
</tr>
<tr>
<td><strong>Costumização</strong></td>
<td>Ação de criar dispositivos para tomar o funcionamento de um programa mais adequado às necessidades pessoais de um usuário ou um grupo de usuários.</td>
</tr>
<tr>
<td><strong>ASCII</strong></td>
<td>Formato de arquivo baseado em um conjunto determinado de caracteres.</td>
</tr>
<tr>
<td><strong>HTML</strong></td>
<td>Linguagem para criação de páginas para a Internet.</td>
</tr>
</tbody>
</table>

Figura 1.5-1 – Convenções e termos técnicos.
1.6 - Resumo do Conteúdo por Capítulo

Pode-se resumir o conteúdo desta Dissertação em 8 Capítulos (Figura 1.6-1).

<table>
<thead>
<tr>
<th>1) INTRODUÇÃO</th>
<th>Introdução aos conceitos e objetivos envolvidos no projeto e uma visão geral sobre o setor de engenharia estrutural, computação gráfica e modelamento.</th>
</tr>
</thead>
<tbody>
<tr>
<td>2) VIABILIZAÇÃO DO PROJETO</td>
<td>Descrição da análise de mercado, tendências de desenvolvimento, estudo das ferramentas utilizadas no projeto. Conceitos de programação aplicados, interfaces e sistemas utilizados.</td>
</tr>
<tr>
<td>3) FLUXOGRAMA DO PROJETO</td>
<td>Descrição do escopo de desenvolvimento detalhando as etapas do programa e o caráter paralelo do desenvolvimento em equipe.</td>
</tr>
</tbody>
</table>
5) MODELADOR
Descrição do Modelador em sua versão 2.0, manual de utilização, guia do usuário e limitações do programa.

6) CONCLUSÕES E PROPOSTAS PARA TRABALHOS FUTUROS
Conclusões obtidas a partir do desenvolvimento do Modelador, análise dos impactos e benefícios da computação gráfica. Novas implementações a serem realizadas no Modelador. Continuação do estudo e desenvolvimento de projetos utilizando a computação gráfica através do AutoCAD.

7) REFERÊNCIAS BIBLIOGRÁFICAS
Referências bibliográficas utilizadas no projeto tais como: bibliografia impressa, manuais e documentação On-line, suporte através da Internet.

8) APÊNDICES

Figura 1.6-1 – Resumo do conteúdo por capítulo.

1.7 - Objetivos

Na atualidade, muito se tem debatido sobre ganhos e retornos de investimentos em novas tecnologias. O avanço computacional aumenta a cada dia e com ele surgem questionamentos e análises sobre sua aplicação em diversas áreas, inclusive na engenharia.

Este trabalho tem como principal meta mostrar caminhos alternativos para a aplicação da computação gráfica na engenharia minimizando custos e
aumentando a produtividade, qualidade e confiabilidade dos resultados. Partindo-se destes objetivos, este projeto busca aplicar algumas das tecnologias já consagradas a nível mundial e colocadas como padrão de qualidade e eficiência. Especificamente na tecnologia de sistemas CAD, pode-se destacar o AutoCAD como líder mundial em popularidade, flexibilidade e eficiência na aplicação dos conceitos da computação gráfica na área de projetos. Na tecnologia de programação, tem-se atualmente uma mudança de filosofia para a orientação a objetos ou programação orientada a objetos. Esta tecnologia permite um nível muito mais elevado e de maior qualidade na abstração da realidade para o ambiente computacional.

A aplicação dos conceitos de programação orientada a objetos na computação gráfica permite uma maior aproximação da representação da estrutura no ambiente computacional. Utilizando o AutoCAD como plataforma gráfica e a programação orientada a objetos como forma de costumização do AutoCAD para adequá-lo a projetos estruturais de engenharia, este projeto tem como meta a criação de um pré-processador gráfico dentro do próprio AutoCAD permitindo o modelamento de estruturas baseadas em nós, barras e superfícies. Este modelamento permitirá a substituição de um modelamento numérico por um modelamento gráfico usufruindo das vantagens da computação gráfica e da grande flexibilidade e reaproveitamento de dados.

Com base em todo o material teórico e prático obtido com o estudo da computação gráfica aplicada à engenharia, pode-se estabelecer os objetivos globais desta Dissertação. Como comentado anteriormente, toda a proposta de desenvolvimento baseia-se numa pesquisa tecnológica de mercado levando-se em conta todas as novas tendências de desenvolvimento da computação e a evolução da engenharia de estruturas.

O desenvolvimento objetiva atingir e criar uma base que possa oferecer suporte e um ponto de partida para outros desenvolvimentos do mesmo caráter baseando-se na utilização de uma linguagem de programação moderna e de
uma plataforma bastante flexível. Toda esta base serve de estímulo para uma ampliação da aplicação dos conceitos de computação gráfica abrindo-se assim um grande leque de possibilidades mais ousadas e de caráter diversificado, ou seja, este desenvolvimento objetiva tornar-se um ponto de partida para a aplicação da computação gráfica em diversas áreas da engenharia e outros setores tecnológicos.

O caráter deste desenvolvimento não se limita à aplicação específica do desenvolvimento da computação gráfica utilizando a plataforma de desenvolvimento ObjectARX (linguagem C++) e o aplicativo AutoCAD R14. Antes de tudo o foco principal deste trabalho é mostrar as vantagens da computação gráfica aplicada a sistemas e procedimentos que se baseiam em processos repetitivos.

Ao final do projeto, estará concretizado um produto em uma versão inicial que será o resultado do estudo e da aplicação destes conceitos à engenharia de estruturas. A aplicação do desenvolvimento é imediata para as áreas que possuam características similares à engenharia de estruturas, tais como: engenharia elétrica, mecânica, de transportes e muitas outras. A adaptação do desenvolvimento para modelamentos utilizando elementos finitos também é possível pois o projeto baseou-se em um estrutura formada por nós, barras e superfícies. Esta adaptação pode ser feita através da redefinição do comportamento destes três elementos gráficos em função do modelamento em elementos finitos e em função da especificação do elemento finito a ser utilizado. A grande flexibilidade da programação orientada a objetos permite a estruturação e arquitetura totalmente aberta do programa facilitando sua atualização e adaptação a situações específicas.
Capítulo 2

ESTRUTURAÇÃO DOS RECURSOS PARA O PROJETO

Mercado e Tendências Tecnológicas
Metodologia de Projetos de Engenharia Estrutural
Otimização Utilizando a Computação Gráfica
Ferramentas da Computação Gráfica
Plataformas: AutoCAD R14 e Windows NT 4.0
Interfaces Visuais
Programação Orientada a Objetos (POO)
Visual C++ 5.0 e ObjectARX 2.0
CADTEC
CODEME Engenharia Ltda.
2.1 – Mercado e Tendências Tecnológicas

Após o estabelecimento das propostas e metas do projeto, tornou-se necessária a efetivação de uma pesquisa mercadológica para detecção de técnicas e tendências existentes e em uso no mercado de engenharia de estruturas e computação.

Na atualidade, o mercado de engenharia encontra-se em meio a uma grande diversidade de propostas de desenvolvimento e de aplicativos para problemas da prática principalmente no setor de projetos. A partir do conhecimento desta diversidade de tendências, foi de grande importância realizar uma pesquisa entre as empresas e escritórios de projeto estrutural para se conhecer melhor estas novas propostas e tendências.

Foram organizadas visitas de caráter técnico a empresas onde os principais objetivos eram a detecção de carências nos diversos setores e um levantamento mais detalhado das novas técnicas computacionais em andamento.

Os dados colhidos durante as visitas mostraram também uma grande diversidade entre as empresas nos níveis de demanda e, principalmente, no estágio de desenvolvimento do processo de automação de projetos. Em alguns casos específicos, observou-se claramente a existência de pontos críticos no processo produtivo de projetos estruturais causados pela pouca ou mínima aplicação de novas tecnologias de computação e métodos tecnológicos.

Observou-se ainda uma certa insegurança e incerteza em relação às tendências tecnológicas disponíveis e oferecidas no mercado. Algumas empresas optaram pela aquisição de aplicativos comerciais genéricos que, normalmente, não atendem completamente às suas necessidades e, muitas vezes, não justificam o alto investimento. Por outro lado, as empresas que preferiram desenvolver os próprios aplicativos, nem sempre utilizam as
ferramentas mais adequadas ou têm recursos humanos disponíveis para tal fim. Das duas opções, no entanto, ficou relativamente clara a preferência pelo desenvolvimento dos próprios aplicativos que, em geral, trazem melhorias de eficiência e produtividade, além de atender melhor às necessidades da empresa.

Neste ponto ficou bastante evidente o papel da Universidade e a importância da proposta do presente trabalho de estabelecer parcerias com empresas para atender a estas demandas.

A idéia, baseada num esforço bilateral de troca de conhecimento e informação, era de promover o desenvolvimento tecnológico voltado para alguma necessidade específica da empresa, através da pesquisa na Universidade.

A começar por um levantamento dos melhores meios e ferramentas a serem utilizados, a proposta desta pesquisa foi o estudo dos conceitos e metodologias de projetos estruturais com o objetivo de criar um novo ambiente de desenvolvimento de projeto otimizado pela "costumização" e automação.

2.2 – Metodologia de Projetos de Engenharia Estrutural

Analisando a sistematização e a estruturação da metodologia de projetos estruturais na engenharia civil, pode-se chegar a um escopo e fluxograma de procedimentos. Obviamente, este processo apresenta variações dependendo das características de cada empresa assim como do nível de automatização e de aplicações dos conceitos computacionais no processo. Basicamente, pode-se representar o processo de desenvolvimento de projetos estruturais em engenharia conforme mostra a Figura 2.2-1:
Observando o fluxograma do processo, destacando na Figura 2.2-1 o maior retângulo, podemos concluir que este é o ciclo de atividades que demanda o maior fluxo de informações e interdependência entre os processos de engenharia. Nesta fase, o processo exibe uma interdependência importante entre Projeto, Cálculo e o Detalhamento. Muitas vezes é necessário efetuar ajustes no projeto em função de procedimentos de detalhamento, de cálculo ou até mesmo de aspectos arquitetônicos e construtivos. Durante estes ajustes, o tempo é o fator primordial de comprometimento da eficiência de todo o processo.

Mais especificamente, dois pontos são considerados como críticos neste processo. O primeiro deles está na interface entre o projeto conceitual (arquitetônico) e o projeto estrutural. Atualmente, a maioria dos projetos arquitetônicos têm sido feitos através de aplicativos CAD, tais como AutoCAD e Microstation, enquanto os projetos estruturais em geral apresentam padrões
diversos. A complexidade e as dificuldades de desenvolver uma interface para tradução e compatibilização do formato do projeto arquitetônico e da padronização da empresa de projeto estrutural, acabaram adiando ou até inviabilizando tal iniciativa. Muitas vezes é necessário quase refazer o projeto para torná-lo compatível com os sistemas de engenharia estrutural o que, freqüentemente, representa um ponto de estrangulamento do processo.

O segundo ponto crítico aparece na fase de detalhamento da estrutura. Muitos dos aplicativos comerciais disponíveis não oferecem recursos satisfatórios para o detalhamento, tornando necessária uma conferência e revisão minuciosa do detalhamento gerado pelo programa de cálculo para atender aos padrões da empresa e às melhorias dos aspectos construtivos.

Uma das propostas deste trabalho é o de dar início a um processo de automatização total de projetos estruturais, desde a concepção até a manufatura. Neste sentido, o Modelador Estrutural poderia ser aplicado na definição da estrutura a partir do projeto arquitetônico, garantindo a compatibilidade entre os dois projetos.

2.3 – Otimização Utilizando a Computação Gráfica

Após a identificação das etapas críticas do processo, pode-se estabelecer diretrizes para propor e implementar procedimentos computacionais que acelerem estas etapas do processo.

A interface projeto arquitetônico / estrutural é responsável pela tradução do projeto fornecido pela arquitetura para o projeto estrutural. Geralmente, a entrada de dados nos aplicativos de cálculo é feita através de um processamento gráfico próprio ou de processamento numérico de dados
(Figura 2.3-1). Provavelmente, o usuário sentir-se-ia mais a vontade se pudesse modelar a estrutura no mesmo ambiente do projeto arquitetônico.

![Diagrama com nós e barras](image)

<table>
<thead>
<tr>
<th>Nó</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Barra</th>
<th>Nó I</th>
<th>Nó F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>1</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5.0</td>
<td>0.0</td>
<td>0.0</td>
<td>2</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>0.0</td>
<td>5.0</td>
<td>0.0</td>
<td>3</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>5.0</td>
<td>5.0</td>
<td>0.0</td>
<td>4</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>0.0</td>
<td>0.0</td>
<td>3.0</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>5.0</td>
<td>0.0</td>
<td>3.0</td>
<td>6</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>0.0</td>
<td>5.0</td>
<td>3.0</td>
<td>7</td>
<td>6</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>5.0</td>
<td>5.0</td>
<td>3.0</td>
<td>8</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**Figura 2.3-1 – Comparação entre interface gráfica e numérica.**

Uma das propostas do presente desenvolvimento do Modelador Estrutural é de criar uma interface onde o engenheiro estrutural possa modelar sua estrutura utilizando o mesmo padrão dos arquitetos. Desta forma, o engenheiro não ficaria preso à formatação rígida dos números e poderia contar com a flexibilidade de uma interface gráfica definida na arquitetura.

Após o modelamento, o engenheiro poderia armazenar os dados da estrutura juntamente com os dados do projeto arquitetônico e, posteriormente, este modelamento estaria disponível para edições como desejado. Para tornar esta interface mais amigável ainda, o Modelador poderia exportar todos os dados geométricos para qualquer programa de análise que pudesse ser alimentado com dados via arquivo texto (ASCII).

Com a implementação desta interface, os ganhos em velocidade e flexibilidade seriam substanciais podendo assim minimizar os efeitos do primeiro ponto de estrangulamento do processo de projetos estruturais. Um
fato importante a ser mencionado, é a flexibilidade do aplicativo AutoCAD a ser utilizado, considerando ser esta uma interface gráfica bastante difundida no meio da engenharia e arquitetura. Como resultado, tem-se a possibilidade de minimizar a etapa de treinamento dos usuários. Como poderá ser visto no item 2.8, o Modelador utilizou ferramentas que possibilitam um funcionamento bastante similar ao do AutoCAD.

Devido à complexidade e importância desta primeira interface, os estudos e desenvolvimentos desta dissertação vão se limitar a este primeiro ponto crítico ficando o segundo, a etapa de detalhamentos, para trabalhos futuros.

2.4 – Ferramentas da Computação Gráfica

Estabelecidas as metas iniciais de desenvolvimento e implementação do Modelador, o próximo passo foi a escolha da plataforma gráfica a ser utilizada.

Uma boa plataforma deveria oferecer algumas características que pudessem tornar o Modelador Estrutural mais simples de utilizar, mais flexível e com sua arquitetura programática o mais aberta possível. Analisando os aplicativos em uso corrente no mercado de engenharia e arquitetura, um destes se destacou perante os outros, que foi o AutoCAD.

Na época, disponível em sua versão 13, já era um aplicativo bastante consolidado e, certamente, o mais utilizado no meio (cerca de 70% do mercado mundial segundo o fabricante, a Autodesk). O AutoCAD já vinha sendo também utilizado na própria Escola de Engenharia como o aplicativo padrão das novas disciplinas de desenho.

Neste caso, o primeiro requisito, o de ser uma plataforma bastante difundida entre os engenheiros e arquitetos, já estaria cumprido. Outra
característica de fundamental importância é sua arquitetura aberta para programação, especialmente a versão 14, que permite o desenvolvimento de aplicativos para áreas específicas.

A arquitetura do AutoCAD R13 (versão 13) oferece três possibilidades de programação; AutoLISP, ADS e ARX, o que inicialmente era um fator bastante animador, pela possibilidade de escolher a linguagem mais adequada. Durante os estudos das potencialidades do AutoCAD R13, a versão 14 foi disponibilizada no mercado americano, com vários recursos gráficos novos e uma arquitetura ainda mais aberta para a programação.

2.5 – Plataformas: AutoCAD R14 e Windows NT 4.0

Com a nova versão do AutoCAD, foram disponibilizadas algumas opções a mais de linguagens para desenvolvimento: AutoLISP, VisualLISP, ADS, ADSRX, ObjectARX (antigo ARX), Visual Basic, Active-x. A nova interface gráfica HEIDI oferece uma manipulação extremamente rápida dos desenhos requerendo a mesma configuração de equipamento que o usuário já possuía na versão 13. Estas novas linguagens disponíveis para desenvolvimento mostram-se totalmente compatíveis com as novas tendências de programação do campo da computação utilizando, em alguns casos, técnicas modernas de orientação a objetos.

A arquitetura flexível do AutoCAD permite que sejam desenvolvidos aplicativos complexos com módulos desenvolvidos em linguagens diferentes. Isto deve-se à aplicação e utilização dos conceitos de programação COM (Component Object Model) que permite a comunicação entre módulos de programação utilizando os conceitos de programação orientada a objetos [Kru97]. Esta estruturação torna possível a construção de um aplicativo base e a criação de extensões, utilizando outras linguagens de programação no
AutoCAD, que se utilizem deste aplicativo base através da interface COM. Dessa forma, torna-se possível a utilização de módulos gráficos em VBA (Visual Basic for Applications), rotinas complexas em ObjectARX (C++) e funções simples de "costumização" utilizando Active-X.

Todos estes fatores e outros mais determinaram a escolha do AutoCAD R14 como a plataforma gráfica para o desenvolvimento do Modelador Estrutural. A versão 14 foi disponibilizada nas plataformas Windows 95 e NT para PC. Seguindo uma tendência em desenvolvimentos foi adotada a plataforma Windows NT 4.0.

A plataforma Windows NT 4.0 mostrou-se muito mais estável quanto ao desempenho dos aplicativos utilizados em conjunto, uma maior flexibilidade na execução dos programas e possibilidade de controle sobre a ocorrência de erros de execução. Um outro fator decisivo é que a plataforma Windows NT 4.0 oferece uma maior atualização de aplicativos e é totalmente compatível com o funcionamento padrão do Windows 95.

Aplicativos desenvolvidos na plataforma Win32 (utilizando 32 bits) funcionam em ambos os sistemas Windows NT 4.0 e Windows 95. Uma pequena desvantagem no uso da plataforma em Windows NT 4.0 é a necessidade de maior disponibilidade de memória RAM. Para o Windows 95, o recomendado é 32 Megabytes enquanto que, no Windows NT 4.0, o recomendado é 48 Megabytes de memória RAM.

A princípio, estas características das plataformas de sistemas operacionais podem parecer de pouca importância para o desempenho do desenvolvimento mas, a longo prazo, a estabilidade do sistema operacional é essencial ao processo. O Windows NT 4.0 mostrou-se muito mais estável e seguro durante a execução de vários aplicativos isoladamente ou em conjunto, o que levou à sua adoção para o desenvolvimento do projeto Modelador juntamente com o AutoCAD R14.
2.6 – Interfaces Visuais

Novos conceitos de programação têm sido introduzidos e aperfeiçoados como simples mas importante objetivo de tornar a interação do usuário com o computador cada vez mais amigável e agradável. Os sistemas operacionais baseados em janelas (sistemas Windows) substituíram aqueles baseados em telas de texto com grande ganho de produtividade, além de ter trivializado a comunicação do usuário com o computador. A Figura 2.6-1 ilustra a potencialidade de comunicação visual dos dois tipos de sistemas operacionais.

![Tela Texto](image1)

![Tela Windows](image2)

Figura 2.6-1 – Modelos de interfaces em modo Texto e modo Gráfico.

A evolução dos sistemas operacionais ocorreu simultaneamente com a computação gráfica resultando no aparecimento de diversos aplicativos gráficos amigáveis e com uma nova forma de comunicação visual com o usuário através de quadros de diálogo e barras de ferramentas. Mais recentemente, outros dispositivos vêm contribuindo para melhoria desta interface tais como: “tooltips” (dicas), “wizards” (assistentes) e dispositivos de ajuda “on-line”.

Seguindo esta tendência de interfaces visuais, o presente Modelador adotou o mesmo padrão que é do próprio AutoCAD, desde a sua versão 12, para sistema Windows e, mais recentemente, a versão 14 foi lançada exclusivamente para a plataforma Windows em computadores do tipo PC.

Utilizando algumas das mesmas ferramentas utilizadas para desenvolver os aplicativos já consagrados e difundidos mundialmente no sistema Windows, o Modelador é um programa que utiliza o tipo de interface adotada como padrão pelos programadores. A integração do Modelador ao ambiente Windows torna-se facilitada pelo desenvolvimento através dispositivos gráficos de comunicação com o usuário similares ao próprio sistema Windows (quadros de diálogo, barras de ferramentas, dicas, etc.). Esta ambientação facilita ainda mais a ambientação do usuário com o novo aplicativo e seu rápido aprendizado baseado nos mesmos conceitos já estudados.

A partir desta interface o Modelador foi projetado para ser um programa amigável e de fácil utilização. Espera-se que o usuário sinta-se a vontade para explorar os comandos do Modelador e consiga atingir rapidamente um ótimo uso do aplicativo e de suas interfaces.

2.7 – Programação Orientada a Objetos (POO)

Juntamente com as novas técnicas computacionais, surgiram novos conceitos de programação que extrapolam a própria codificação da linguagem e atingem outros níveis de abstração.

A programação orientada a objetos (POO) surgiu de uma analogia ao desenvolvimento da indústria de “Hardware” onde o reaproveitamento de esforços foi o principal motivo de um progresso bastante significativo. A estrutura dos computadores passou a ser formada por partes padronizadas,
baratas e testadas por diversos usuários a nível mundial e fabricadas por várias empresas [Mon94]. Na indústria de “Software”, não ocorria algo semelhante pois a reutilização de código era baixa e a criação repetitiva de rotinas era constante.

O objetivo era então estruturar sistemas de grande porte baseados em sistemas menores e modularizados de uma maneira mais fácil e eficiente do que a programação estruturada. Historicamente, estes princípios surgiram do amadurecimento da linguagem Simula, na década de 60, e reimplementados na década de 70, na linguagem Smalltalk. A partir da década de 80, estes conceitos encontram-se incorporados na linguagem C++ [Mon94].

Para possibilitar um melhor aproveitamento de código foi criada na década de 80, por Bjarne Stroustrup, nos laboratórios da AT&T, a linguagem C++ com o objetivo de integrar os conceitos da POO, presentes em Simula, à linguagem C. O C++ foi projetado a partir de uma abordagem de Simula utilizando a linguagem C, uma das mais populares, como base [Ell93].

A comunicação entre os procedimentos diferencia basicamente os enfoques de programação estruturada e programação orientada a objetos. Na programação estruturada os procedimentos são implementados em blocos que se intercomunicam através da passagem de parâmetros. Estes procedimentos processam os dados e estes são passados de um procedimento para outro. O programa é então caracterizado pelo acionamento destes procedimentos e pela troca de dados entre eles. Na programação orientada a objetos os procedimentos e dados fazem parte de somente um elemento básico (objeto ou classe). Estes elementos comunicam-se caracterizando a execução do programa (Figura 2.7-1). Na POO, os dados e procedimentos estão encapsulados em um só elemento [Mon94].
Esta estrutura torna o aplicativo muito mais flexível e muito menos sensível a alterações nos dados ou em funções específicas. Com a adoção da POO pelos desenvolvedores de aplicativos e sistemas de grande porte, surgiu uma nova filosofia de desenvolvimento de “Software”. Com a POO, é possível criar aplicativos com uma arquitetura aberta tornando possível a inclusão de novas rotinas e funções dentro dos próprios programas ou acopladas a eles (Figura 2.7-2). Dessa forma, os grandes desenvolvedores de “Software” passaram a priorizar muito mais a eficiência dos programas. Pode-se dizer que hoje existe uma tendência de ampliação do potencial desta nova forma de “costumização” de aplicativos. No caso específico do AutoCAD, o ambiente de programação ObjectARX permite a aplicação dos conceitos de POO através da linguagem C++.

Outro ponto importante na POO é a grande facilidade de desenvolvimento de trabalhos em equipe. Através da abstração eficiente dos dados, pode-se criar uma estrutura básica do aplicativo a ser desenvolvido e
assim permitir que as novas implementações sejam feitas de maneira simples e organizada. As dificuldades iniciais na utilização da POO são perceptíveis em relação à programação estruturada mas se justificam plenamente com o passar do tempo, em vista às facilidades e ganhos em produtividade e desempenho do aplicativo.

Figura 2.7-2 – Estruturação de um aplicativo utilizando a POO.

2.8 – Visual C++ 5.0 e ObjectARX 2.0

Para o desenvolvimento do presente projeto, foram escolhidos os ambientes de desenvolvimento Visual C++ 5.0 e ObjectARX 2.02. A escolha do compilador, o Visual C++ 5.0, foi a única alternativa predefinida pela Autodesk, em parceria com a Microsoft, a partir da versão 13 do AutoCAD. Na versão 13 do AutoCAD, foi adotada a versão 4.0 do Visual C++ e, a partir da versão 14, foi oferecido suporte para versões 4.2b e 5.0 do Visual C++.
Como mencionado no item 2.5, o AutoCAD, em sua versão 14, oferece diversas possibilidades de programação porém a mais avançada e com maior potencial é o ambiente ObjectARX. O ObjectARX, atualmente em sua versão 2.02, possibilita a aplicação dos conceitos de POO e dos recursos da linguagem C++.

O ObjectARX é um conjunto de classes e métodos disponibilizados a partir da criação do AutoCAD, ou seja, são oferecidas aos desenvolvedores algumas das bibliotecas utilizadas na programação e desenvolvimento do próprio AutoCAD. Todas estas facilidades possibilitam um desempenho equivalente ao dos procedimentos nativos do AutoCAD. Em resumo, os aplicativos implementados através da linguagem C++, utilizando o ambiente ObjectARX, são executados como se fossem extensões do AutoCAD.

Todos estes fatores, aliados ao potencial da linguagem orientada a objetos C++, determinaram a escolha deste ambiente de desenvolvimento para este trabalho. Através do Visual C++ 5.0, pode-se criar um aplicativo completo utilizando o ambiente ObjectARX, ou seja, um aplicativo com interfaces visuais, barras de ferramentas e rotinas gráficas. Como o Visual C++ tem sua base de programação com suporte às ferramentas do Windows (MFC), todos estes recursos podem também ser utilizados pelo ObjectARX.

Além disso, o potencial do ObjectARX permite que os aplicativos possam ser criados em versões sucessivas sem as possíveis perdas de compatibilidade. Dessa forma, pode-se criar aplicativos que têm capacidade para manipular arquivos de versões anteriores. Todo o mecanismo do ObjectARX permite um controle total do comportamento do AutoCAD quando o aplicativo ObjectARX está sendo executado.

Através da estruturação da POO e da linguagem C++, as bibliotecas do ObjectARX são disponibilizadas para possíveis derivações entre classes nativas do AutoCAD. Dessa forma, novas entidades podem ser criadas.
(recurso este possível somente na utilização do ObjectARX na “costumização” do AutoCAD) e terão seu comportamento determinado e gerenciado pelo próprio AutoCAD.

Outra grande vantagem na utilização do ObjectARX é a possibilidade de manutenção do banco de dados do próprio AutoCAD. Assim, é possível o controle total sobre as entidades nativas e as “costumizadas” criadas pelo aplicativo. Além disso, como o ObjectARX é uma biblioteca na linguagem C++, pode-se utilizar, quase que em sua totalidade, as classes e métodos do C++. A grande vantagem desta característica é a possibilidade de acesso a funções matemáticas avançadas, recursos gráficos, manipulação de arquivos externos, banco de dados, etc.

Um ponto negativo a ser apontado é a falta de documentação e suporte ao ObjectARX. Lançado efetivamente juntamente com a versão 14 do AutoCAD, o ambiente ObjectARX começou a ser difundido nos Estados Unidos no final de 1996. Em Maio de 1997 o AutoCAD R14 foi lançado nos Estados Unidos juntamente com o ObjectARX 2.02, quando haviam duas opções para o desenvolvimento; a plataforma AutoCAD R13 e o ambiente ObjectARX 1.1 e o AutoCAD R14 e ObjectARX 2.02.

Outro ponto negativo a ser ressaltado é a necessidade de atualização dos aplicativos desenvolvidos em ObjectARX quando novas versões do AutoCAD forem lançadas. Para cada versão do AutoCAD colocada no mercado a Autodesk lança paralelamente uma versão correspondente do ObjectARX. Esta atualização do código é feita através de uma recompilação do código incluindo as novas bibliotecas lançadas. A filosofia de desenvolvimento da Autodesk mantém a base programática quase inalterada. As melhorias são quase sempre efetuadas acrescentando novas classes e disponibilizando novas funções. O principal sinal desta filosofia de desenvolvimento é a possibilidade de abrir arquivos do AutoCAD de versões anteriores, por exemplo a versão 2.5, em versões atuais como a 14.
Durante o processo de análise das opções disponíveis para o desenvolvimento, foi efetivada uma parceria entre o CADTEC e a Autodesk através de sua rede mundial de desenvolvedores (ADN – Autodesk Developer Network). Com a parceria, os desenvolvedores envolvidos no projeto do Modelador Estrutural passaram a ter acesso a uma série de dispositivos de apoio para o desenvolvimento. Como membro ADN, o CADTEC passou a receber a coleção completa e atualizada dos programas e aplicativos desenvolvidos pela Autodesk acompanhados de documentação técnica, além de assistência técnica direta por telefone, fax ou e-mail. No caso do suporte via e-mail, uma equipe mundial de consultores fica disponível para a resolução de dúvidas e problemas relacionados com os desenvolvimentos em ObjectARX.

Apesar das incertezas e dificuldades em se adotar uma plataforma recém lançada, dada a literatura limitada disponível, o AutoCAD R14 e o ObjectARX 2.0 foram adotados com sucesso.

2.9 – CADTEC

A concepção do projeto Modelador iniciou-se juntamente com a construção de um espaço físico que viabilizasse o seu desenvolvimento. Assim, foi fundado o CADTEC (Centro de Apoio, Desenvolvimento Tecnológico e Ensino da Computação Gráfica) cujo papel tem sido preponderante no apoio técnico, no treinamento e na disponibilização de recursos para desenvolvimentos na área da computação gráfica.

O CADTEC é um Centro do Departamento de Engenharia de Estruturas da UFMG e teve como seu projeto piloto, o Modelador Estrutural. Durante o último ano vários novos integrantes aliaram-se aos objetivos do CADTEC de promover a aplicação da computação gráfica a projetos.
O grupo CADTEC consiste hoje de cerca de dez integrantes entre professores, alunos de graduação e de pós-graduação. Dentro de um caráter multidisciplinar, as propostas do CADTEC procuram promover a participação de outros departamentos e unidades da UFMG e de outras universidades. Além disso, o CADTEC tem procurado desenvolver seus projetos em parceria com a iniciativa privada para que problemas práticos possam ser os principais objetivos de estudo.

2.10 – CODEME Engenharia Ltda.

Os primeiros oito meses do desenvolvimento do Modelador Estrutural foram feitos em parceria com a empresa CODEME Engenharia Ltda. A participação da CODEME foi fundamental na definição do escopo do projeto e de importantes aspectos técnicos do desenvolvimento. A equipe técnica da CODEME participou e contribuiu mais significativamente ainda nos aspectos relacionados com sua grande experiência nos conceitos de engenharia e de desenvolvimento automatizado de projetos estruturais.

Embora o Modelador Estrutural apresente algumas características específicas e peculiares do processo de automatização de projetos da CODEME, o desenvolvimento serviu para estabelecer alguns padrões necessários em um bom modelador de estruturas.

Isto foi possível com a participação efetiva da CODEME com sugestões, avaliações e críticas construtivas dos desenvolvimentos. Através de reuniões periódicas entre a equipe de desenvolvedores do CADTEC e a equipe técnica da CODEME, foram traçadas as diretrizes iniciais do projeto assim como foram feitas avaliações de etapas alcançadas, erros e modificações aplicáveis ao projeto.
Capítulo 3

FLUXOGRAMA DO PROJETO

Escopo Conceitual
Desenvolvimento Paralelo
Fluxograma do Projeto
3.1 - Escopo Conceitual

Com a efetivação da parceria entre o CADTEC e a CODEME Engenharia Ltda., a escolha da linguagem C++, do ambiente ObjectARX plataforma de desenvolvimento AutoCAD, a etapa seguinte do projeto foi o estabelecimento de um escopo conceitual sobre o desenvolvimento do projeto.

Pela própria característica da POO, a organização do desenvolvimento é bastante facilitada e direcionada para uma formatação mais flexível e independente. Assim, a divisão do projeto em partes foi essencial e, de certa forma, imediata. Toda a base do desenvolvimento em ObjectARX é feita através de entidades e objetos onde a diferença básica entre estes dois é, no caso da entidade, a presença de uma representação gráfica.

Para se abstrair os dados de uma estrutura pode-se determinar a criação de três entidades (objetos) básicos:

- **Nós** (conexões entre Barras e pontos referenciais para ligações e aplicação de carregamentos);
- **Barras** (ligação entre Nó obedecendo a incidência entre ponto inicial e final proporcionando a organização e o modelamento da estrutura em forma de malha unifilar);
- **Faces** (região delimitada por barras que representa uma superfície que pode receber carregamentos e tem comportamento estrutural determinado);

Estes elementos determinam um dos tipos de processamento de uma estrutura, que podem variar de acordo com o método utilizado para análise e o padrão de trabalho adotado. A grande vantagem da aplicação dos conceitos da POO é a facilidade da abstração dos elementos estruturais em forma de objetos, ou seja, em nó, barra e face.
A partir deste princípio, a organização conceitual do desenvolvimento do projeto foi estabelecida em uma sequência lógica e dinâmica visando estruturar o desenvolvimento em etapas sucessivas utilizando uma formatação contínua e flexível. As etapas podem então ser divididas sob o ponto de vista conceitual (Figura 3.1-1).

Figura 3.1-1 – Fluxograma do Escopo Conceitual.
Esta organização permite um correto direcionamento do desenvolvimento no que diz respeito à aplicação dos conceitos de POO. O encapsulamento dos dados em cada elemento estrutural permite que outros dados sejam inseridos em etapas seguintes de forma simplificada e segura. A estrutura funcional fica então, do ponto de vista conceitual, determinada na manipulação dos três elementos estruturais básicos e em suas respectivas implicações e implementações.

Pensando em manter a arquitetura do aplicativo a mais aberta possível, foi adotado o armazenamento de dados nas próprias entidades, ou seja, somente serão usadas as entidades geométricas para representar e conter os dados relativos aos elementos estruturais. Dessa forma, os carregamentos, por exemplo, serão armazenados nas próprias entidades estruturais e não será necessária a criação de novas entidades exclusivamente para armazenar os dados de carregamentos.

Outra vantagem da estruturação neste formato é a possibilidade de aplicar novos métodos em cada entidade com o desenvolvimento do aplicativo, ou seja, retornar facilmente a etapas anteriores e acrescentar dados ou funções nas entidades. Como a estruturação está baseada nestes elementos, as alterações intrínsecas não causarão, a princípio, mudanças bruscas no funcionamento geral do aplicativo. Muito pouco além de pequenos ajustes compatíveis com as novas implementações serão necessários durante o desenvolvimento.

3.2 - Desenvolvimento Paralelo

A grande dimensão, a complexidade do projeto e sua meta de criar um Modelador Estrutural de caráter inovador, levando-se em conta a interface AutoCAD, levou à divisão do projeto em duas partes paralelas de
desenvolvimento para se atingir de forma mais rápida e eficiente, os objetivos do projeto.

Cada parte está direcionada a um pesquisador integrante do projeto. Esta divisão foi viabilizada pela flexibilidade da POO e pela versatilidade da estrutura proposta para este aplicativo. A manutenção do desenvolvimento, de forma a manter ambas as partes em perfeita sincronia e caminhando lado a lado, é essencial uma vez que os resultados obtidos em cada etapa servem como base para as etapas seguintes e para melhorias nas implementações sucessivas.

O trabalho em equipe aparece como ponto chave no desenvolvimento e na obtenção de conclusões e respostas rápidas a cada etapa alcançada. Esta evolução paralela permite a detecção de erros de forma mais eficiente evitando assim que ambas as partes possam incorrer na repetição de erros e muito do caráter conceitual pode ser reutilizado e aprimorado.

Além do controle conceitual do desenvolvimento paralelo um ponto relevante e de muita importância é o da manutenção dos arquivos e das implementações efetivadas sobre o programa. Cada mudança deve ser comunicada à equipe e todos os impactos devem ser discutidos e analisados. As novas implementações devem prezar sempre pela padronização seguindo a formatação padronizada do programa contendo, além do código totalmente organizado e comentado, suas versões e data de implementação.

Outro ponto importante é a padronização na criação de rotinas utilitárias. Quando bem estruturadas, podem servir para outros propósitos no programa assim como servir de ponto de partida para rotinas mais complexas que possam ter a mesma base. O caráter genérico deve ser mantido, quando possível, para manter a portabilidade das rotinas e aumentar sua eficiência diminuindo assim o código do programa.
A falta de controle no processo de desenvolvimento paralelo pode gerar problemas de compatibilidade entre os módulos desenvolvidos em separado e causar erros de difícil detecção quando ocorrerem agrupamentos de rotinas. O desenvolvimento paralelo necessita ainda de uma permanente avaliação e da aplicação de testes bastante abrangentes utilizando preferencialmente usuários que não façam parte da equipe de desenvolvimento. Os integrantes da equipe podem ficar com alguma espécie de vício durante a realização dos testes não abrangendo assim um maior número de possibilidades de erros.

O desenvolvimento paralelo, juntamente com a filosofia de programação orientada a objetos, favorece também a facilidade de implementações futuras no aplicativo desenvolvido em face a sua documentação e organização do código. Um programa bem documentado permite a compreensão quase que imediata de um programador externo ao projeto.

Todos estes e outros fatores tornam a estrutura do projeto bastante sólida para servir como base para novas implementações e para o desenvolvimento de outros projetos de igual caráter tomando como base a experiência e os resultados obtidos com o projeto do Modelador Estrutural.

3.3 – Fluxograma do Projeto

A partir da estruturação conceitual do programa e dos conhecimentos de POO, pode-se delinear um fluxograma organizado em etapas para o desenvolvimento do projeto. Cada etapa depende de uma correta implementação das etapas paralelas ou precedentes onde testes deverão ser efetuados de forma isolada para prevenir erros que possam ser repassados para etapas posteriores. As rotinas devem manter o máximo de flexibilidade e usufruir ao máximo da característica de encapsulamento dos dados em cada entidade (objeto).
Inicialmente, os três elementos básicos estruturais deverão ser abstraídos em objetos. Cada objeto deverá ser implementado e testado de forma isolada buscando a eliminação inicial de erros e a previsão de futuros problemas e detecção de novas implementações a serem efetuadas.

**Elaboração do Escopo do Programa**

<table>
<thead>
<tr>
<th>Estruturação Inicial</th>
<th>Implementação Inicial das Entidades: NODE, BAR e a propriedade LAYER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementações Avançadas, Comportamento Conjunto de Nós e Barras, Verificações de Consistência</td>
<td>Análise do Funcionamento das Entidades com Base na Engenharia e Avaliação Geral do Sistema</td>
</tr>
<tr>
<td>Cópia de Entidades, Numeração Automática, Controle de Banco de Dados, Limitações e Ajustes</td>
<td>Implementação da Entidade FACE, Funções de Edição e Implementação Gráfica do Carregamento.</td>
</tr>
<tr>
<td>Casos de Carregamento, Carga Concentrada em Nós, Interface para Carga Distribuída em Superfícies</td>
<td>Implementação da Carga Distribuída na Entidade FACE</td>
</tr>
<tr>
<td>Controle de Visualização</td>
<td>Análise e implementação da Interface Gráfica com o Usuário</td>
</tr>
<tr>
<td>Exportação de Dados</td>
<td>Implementação da Carga Concentrada na Entidade BAR</td>
</tr>
<tr>
<td>Dispositivos Globais Gerenciamento de Módulos</td>
<td>Implementação da Carga Distribuída Parcial / Completa na Entidade BAR</td>
</tr>
</tbody>
</table>

**Análise, Testes e Documentação do Modelador**

Figura 3.2-1 – Fluxograma global do projeto.
Nesta dissertação será implementada e desenvolvida a ramificação destacada no fluxograma global do projeto (Figura 3.2-1) levando-se em conta a evolução do projeto de forma global. Inicialmente, serão realizadas implementações básicas em conjunto para delinear o desenvolvimento das entidades pensando no funcionamento global do programa. A abertura da estrutura do programa deve ser mantida e alterada de acordo com o acréscimo de dados nas entidades básicas.

O fluxograma do presente trabalho foi dividido em etapas de desenvolvimento que determinam o formato geral do Modelador e a sequência lógica das implementações sucessivas. Alguns resultados e conclusões obtidas em cada etapa podem exigir alterações ou novas implementações em etapas anteriores. Cada implementação efetuada com retrocesso no fluxograma deve ser analisada quanto às implicações em outras funções interrelacionadas. Cada entidade deve ter o seu comportamento verificado quanto à consistência dos dados, quanto ao armazenamento de novos dados e quanto à manutenção de sua flexibilidade. A seguir, apresenta-se o fluxograma do presente trabalho de forma mais detalhada destacando-se as principais características e implementações de cada etapa do projeto (Figura 3.2-2).
| Estruturação Inicial, Entidades, Objetos, Banco de Dados do AutoCAD, Testes, Monitoração de Rotinas, Criação de Entidades Básicas. |
| Implementações Avançadas, Comportamento Conjunto de Nós e Barras, Verificações de Consistência em Entidades, Verificações Conjuntas, Operações no Banco de Dados do AutoCAD. |
| Cópia de Entidades, Numeração Automática das Entidades, Controle de Banco de Dados, Limitações ao Usuário, Ajustes Automáticos Após Comandos, Reatores de Entidades, Reatores de Comandos. |
| Casos de Carregamento, Carga Concentrada em Nós, Interface para Carga Distribuída em Superfícies, Remoção de Cargas, Manipulação dos Objetos de Casos de Carregamentos. |
| Controle de Visualização dos Dados das Entidades, Opções de Exibição de Entidades por Tipo ou Numeração, Exibição de Carregamentos. |
| Exportação de Dados, Formatação do Arquivo de Exportação, Interface com Programas de Análise, Coleta de Dados das Entidades. |
| Dispositivos Globais, Módulo Central do Modelador, Gerenciamento de Módulos, Utilização de Barras de Ferramentas. |

Figura 3.2-2 – Fluxograma de desenvolvimento do presente trabalho.
Capítulo 4

DESENVOLVIMENTO DO PROJETO

Estruturação Inicial
Criação de Entidades Básicas
Implementações Avançadas
Carregamentos
Controle de Visualização
Exportação de Dados
Dispositivos Globais
4.1 – Estruturação Inicial

Seguindo os conceitos obtidos a partir da POO e utilizando também as conclusões da elaboração de um escopo conceitual do projeto, pode-se criar uma base de desenvolvimento para o aplicativo.

O principal objetivo desta base é proporcionar uma organização lógica do código assim como proporcionar um desenvolvimento organizado e centrado em apenas um ou dois módulos centrais de gerenciamento. A POO e a estrutura da linguagem C++ permitem e facilitam a programação através de módulos de código que funcionam em conjunto, ou seja, em forma de bibliotecas. Dessa forma, pode-se criar módulos independentes que podem ser carregados em conjunto para formar um aplicativo de maior porte com uma estrutura bastante flexível.

Neste caso, utilizam-se bibliotecas com vínculo dinâmico, as chamadas DLL's (Dynamic Linked Library). Esta formatação de aplicativos permite que todos estes módulos trabalhem de forma independente compartilhando de um mesmo campo de execução na memória do sistema operacional.

Como mencionado anteriormente, os aplicativos desenvolvidos em ObjectARX compartilham do mesmo ambiente de memória que o AutoCAD. O próprio AutoCAD é composto por uma série de módulos que são gerenciados por um módulo executável central (um arquivo EXE). Este módulo principal controla o carregamento e a manipulação dos demais módulos secundários.

Seguindo esta organizaçào funcional, o Modelador também foi estruturado de forma semelhante seguindo e aplicando os conceitos da POO.
4.1.1 – Estrutura Global do Programa

O C++ permite a divisão do aplicativo em módulos utilizando dois tipos de arquivos (com extensões diferentes). O primeiro deles, arquivos com extensão H, são utilizados para declarações de classes e funções. Este arquivo, chamado de arquivo de cabeçalho, contém todos os protótipos de classes, objetos e funções disponibilizadas ao programador. O segundo, arquivos com extensão CPP, são utilizados para definir e implementar os componentes das classes, funções e instâncias de objetos.

Na verdade, o arquivo com extensão H tem o mesmo formato que o arquivo com extensão CPP. A grande diferença, está no tratamento destes arquivos pelo compilador C++. Durante a fase de compilação, o compilador busca todos os arquivos com extensão H e, em seguida, procura as definições de classes e funções nos arquivos CPP, OBJ e LIB. Estes arquivos CPP, quando compilados, se transformam em arquivos OBJ ou LIB. Após esta etapa, o compilador faz a montagem dos diversos arquivos OBJ e LIB gerando então a biblioteca de vínculo dinâmico (DLL). No caso específico do ObjectARX, estas bibliotecas recebem a extensão ARX mas nada mais são do que arquivos do tipo DLL.

Segundo esta formatação, as partes que são montadas para gerar cada arquivo ARX podem e devem ser organizadas em arquivos com extensão H e CPP. Dessa forma, quando se trabalha em equipe, se for necessária a passagem de código entre desenvolvedores, esta deverá ser feita através dos arquivos com extensão H, OBJ e LIB. Quando o integrante do projeto recebe estes dois arquivos, todas as informações necessárias para a utilização deste módulo já estão detalhadas no arquivo com extensão H, arquivo no qual o usuário tem livre acesso. Já os arquivos com extensão OBJ e LIB estão compilados e codificados e o integrante não tem acesso às informações e nem mesmo pode alterar este arquivo.
Toda esta estrutura funciona de forma bastante satisfatória e permite um controle total sobre a segurança dos arquivos. Outra grande vantagem da POO é que, quando os arquivos que contêm definições de classes são alterados e a estrutura de declaração não é alterada, provavelmente o usuário não terá grandes dificuldades para alterar seu programa ou nem mesmo necessitará efetivá-las.

4.1.2 – Reutilização de Rotinas

Como descrito anteriormente, uma das grandes vantagens da POO é o alto nível de reaproveitamento de código. Quando se programa utilizando estes conceitos como base, pode-se construir rotinas com alto nível de complexidade e, totalmente ou em parte, reutilizáveis.

Quando se trabalha com objetos, muitas vezes as operações aplicadas sobre estes objetos são bastante semelhantes e análogas. Seguindo este princípio, todas as rotinas desenvolvidas para o Modelador, quando possível, devem conter uma sintaxe e uma estrutura o máximo possível reutilizável. Como exemplo, pode-se citar uma rotina que adiciona entidades ao banco de dados do AutoCAD. Em particular, esta sequência de chamadas a funções do ObjectARX é um pouco extensa e bastante semelhante para quase todos os tipos de entidades. Dessa forma, a criação de uma rotina que possa efetuar esta sequência para vários tipos de entidades pouparia ao programador um esforço repetitivo e desnecessário. Em contrapartida, o desenvolvimento de uma rotina mais genérica exige um tempo maior de programação mas em compensação, aumenta em muito o ganho no aproveitamento e rendimento do código de forma geral.

No trabalho em equipe, outro ponto importante é o da organização na criação de rotinas, ou seja, quando um dos integrantes desenvolve uma rotina que possa ser útil a outros, esta deve ser distribuída e atualizada de acordo
com a evolução do desenvolvimento e com o aparecimento de novas necessidades.

Utilizando o Visual C++ 5.0, é possível a utilização de Workspaces (ambientes de trabalho). Mais ainda, é possível também a criação de vários projetos dentro de um mesmo Workspace. Cada projeto é composto por um conjunto de arquivos com extensões H, CPP, OBJ, LIB, DEF e outros. Quando se cria um Workspace no Visual C++ 5.0, um diretório é automaticamente criado e nele serão inseridos os projetos.

Cada projeto dá origem a uma biblioteca de vínculo dinâmico (uma DLL). Sendo assim, para cada arquivo ARX gerado deve-se ter um projeto do Visual C++ 5.0 correspondente. Para cada projeto criado dentro de um Workspace é criado também, dentro do mesmo diretório onde o Workspace foi criado, outros diretórios para cada projeto. Assim, os arquivos são organizados e distribuídos nestes diretórios de acordo com sua utilização em cada projeto.

Dentro de um mesmo Workspace, é possível haver um compartilhamento de arquivos como, por exemplo, os arquivos com extensão H e CPP. Arquivos que contenham funções utilitárias podem então ser compartilhados dentro de um mesmo Workspace por vários projetos. Quando estes arquivos compartilhados são alterados, basta que se faça uma recompilação dos projetos que utilizam este arquivo para que as alterações sejam efetivadas corretamente. Assim, mantém-se a compatibilidade entre os vários projetos contidos no mesmo Workspace.

4.1.3 – Implementações Iniciais

O ObjectARX, como um conjunto de bibliotecas escritas na linguagem C++, necessita de um ponto de execução do arquivo principal que executa as demais funções. Este chamado ponto de execução ou ponto de entrada é
descrito na linguagem C++, como uma função particular com o nome MAIN(). Esta função é configurada então pelo compilador como a primeira a ser chamada quando se executa o programa ou a biblioteca.

Como o ObjectARX se destina a compartilhar o mesmo ambiente de trabalho que o AutoCAD utiliza, a forma de execução do arquivo principal é feita através de uma função especial chamada acrXEntryPoint. Esta função deve estar localizada em um dos arquivos com extensão CPP e será aquela pela qual o aplicativo ARX será executado dentro do AutoCAD. Geralmente, este módulo CPP que contém esta função especial recebe o nome de MAIN.ARX e nele se encontram ainda outras operações especiais tais como: registro de comandos, instância de objetos, mensagens de comunicação com o AutoCAD, entre outras.

Basicamente os projetos em ObjectARX resumem-se a um arquivo de entrada (arquivo que contém a função acrXEntryPoint), arquivos com funções utilitárias e arquivos de definição de objetos. Para manter a flexibilidade dos projetos e a modularização do aplicativo, estes arquivos devem ser mantidos organizados e separados por características do seu conteúdo. Assim, devem ser agrupadas funções utilitárias para banco de dados, funções de manipulação de componentes trigonométricos, funções de conversão, etc.

Para se efetivar a montagem de um aplicativo simples, os requisitos básicos são a construção de um projeto com todas as configurações e formatações exigidas pelo ObjectARX, a criação de uma função de execução (ponto de entrada) e a criação de funções de inicialização do aplicativo.

Estas funções de inicialização do aplicativo, ilustradas na Figura 4.1.3-1, podem ser resumidas em:

✓ Ponto de entrada (EntryPoint) : troca de informações com o AutoCAD;
Funções de inicialização tais como: instância (construção) de objetos, construção da hierarquia de classes, inicialização de procedimentos para tratamento de janelas do Windows, etc.;

- Registro de comando: associação de cada comando a uma função específica, armazenamento do comando na pilha de comando definindo nome, grupo e parâmetros adicionais;

- Funções de finalização tais como: destruturas de objetos, descarregamento do aplicativo, etc.;

Figura 4.1.3-1 – Ambiente de execução de um aplicativo em ObjectARX.

4.1.4 – Testes e Monitoração de Rotinas Isoladas

Ao se utilizar a biblioteca ObjectARX, alguns cuidados e procedimentos sequenciais devem ser tomados. Como mencionado em [Aut97a], os recursos de memória do AutoCAD devem ser corretamente gerenciados no espaço
reservado pelo próprio AutoCAD para este fim. Quando se trabalha com objetos, a alocação de memória é constante e necessária. Neste caso específico o próprio ObjectARX redefine ou sobrecarrega as funções e procedimentos que envolvam a manipulação de memória e efetiva as operações de manipulação desta memória no espaço adequado sem conflitos com o espaço ocupado pelo AutoCAD.

Quando se instancia um objeto, é necessária a alocação de uma memória compatível com o tamanho ocupado pelos dados a serem armazenados neste objeto. Quando se exclui um objeto, é necessária a liberação da memória anteriormente ocupada por ele. Os operadores NEW e DELETE, da linguagem C++, efetuam estas alocações e liberações respectivamente. Os objetos utilizados no AutoCAD (entidades ou objetos puros) são armazenados num banco de dados interno do AutoCAD. A partir deste banco de dados, eles são manipulados e armazenados em disco através dos arquivos com extensão DWG (arquivos de desenho do AutoCAD). O próprio AutoCAD decide quando é necessária a criação de um arquivo de paginação em disco para estender a memória do AutoCAD.

Além da alocação e liberação de memória para objetos, é necessário também um gerenciamento da memória quando se utilizam strings (vetores de caracteres) [Mon94]. Estas strings exigem a alocação de um espaço especial e este não pode entrar em conflito com o espaço reservado ao AutoCAD. As funções específicas da linguagem C++ são redefinidas pelo ObjectARX para reservar espaço de memória somente nas regiões que não entrem em conflito com o AutoCAD.

Estes procedimentos especiais foram resumidos em funções utilitárias para facilitar e flexibilizar o desenvolvimento do código do Modelador. Na utilização de janelas (quadros de diálogo), o uso de strings é essencial e realizado frequentemente.
Além dos problemas com alocação de memória, foram encontrados outros detalhes que exigiram a criação de algumas rotinas especiais e mereceram a atenção de alguns testes preliminares.

A seguir, são descritos alguns dos testes realizados numa etapa conjunta entre os integrantes do projeto Modelador:

✓ Função para conversão de pontos no sistema de coordenadas de tela (*ads_point*) para pontos tridimensionais (*AcGePoint3d*);
✓ Função para criação de *Layers* (camadas de desenho);
✓ Função para adicionar entidades ao banco de dados do AutoCAD;
✓ Implementação e teste de instâncias de quadros de diálogo utilizando recursos da *MFC*;
✓ Criação de entidades nativas e entidades “costumizadas”;
✓ Criação de objetos puros;
✓ Manipulação do banco de dados (abertura e fechamento);

Estes testes, aliados a outras experiências e exemplos, contidos no kit de desenvolvimento do ObjectARX, fornecem a base e uma ótima referência para o desenvolvimento do aplicativo, partindo-se de uma base padronizada, testada e confiável. Como os procedimentos no ObjectARX são quase sempre semelhantes, o esforço computacional pode ser bastante reduzido a partir da realização desta etapa preliminar.

4.1.5 – Entidades Nativas e Entidades “Costumizadas”

Quando se trabalha no ambiente ObjectARX, tem-se acesso a quase todos os recursos utilizados pelo AutoCAD. A criação de entidades nativas do
AutoCAD, como linhas, círculos, pontos, arcos, etc., é bastante simplificada e feita de forma bastante eficiente.

Para criar um aplicativo que se utiliza de técnicas e características especiais e particulares, o armazenamento de informações que determinam e gerenciam estas características é estritamente necessário. Estas informações podem conter dados simples ou até vetores complexos.

Especificamente no AutoCAD, existem algumas formas de se armazenar informações especiais além daquelas pertencentes às entidades nativas. Dentre elas, pode-se citar a utilização de dados estendidos (Xrecords), protocolo de extensão (Protocol Extension) e entidades “costumizadas” (Custom Entities). Os dois primeiros podem ser acoplados às entidades nativas do AutoCAD dando assim um acréscimo de informações contidas nestas entidades comuns. Um exemplo bastante simples seria a implementação de características de temperatura à entidade nativa linha (LINE). Ambas as formas citadas possuem uma limitação de armazenamento o que, praticamente, as inviabiliza na construção de aplicativos mais complexos. Além disso, essa limitação acaba por tornar a arquitetura do aplicativo bastante limitada.

Entre as três possibilidades citadas, certamente a utilização de entidades “costumizadas” é a mais atraente e com maior potencial. Uma entidade “costumizada” pode conter características específicas determinadas pelo programador além de ter o seu comportamento gerenciado pelo aplicativo juntamente com o próprio AutoCAD. Numa entidade “costumizada”, pode-se controlar quase que totalmente seu comportamento gráfico, lógico e funcional. Apesar de não ser uma entidade nativa, o AutoCAD, a partir da utilização de sobrecarga de funções, tem controle sobre a entidade e controla várias características especiais para torná-la compatível com o ambiente de trabalho do AutoCAD.
Ao contrário das duas primeiras formas citadas, as entidades “costumizadas” não possuem limite de armazenamento de informações. Além disso, a convivência destas entidades especiais acontece no mesmo ambiente das entidades nativas possibilitando assim a utilização conjunta de ambas. Utilizando-se este meio de armazenamento de dados, pode-se então criar uma funcionalidade totalmente flexível do aplicativo e tornar o comportamento dos elementos a serem utilizados no aplicativo bastante próximos do desejado.

Talvez o único ponto desfavorável para a implementação de entidades “costumizadas” seja o maior nível de complexidade para efetivar esta implementação. As entidades “costumizadas” devem sobrecarregar um conjunto mínimo de funções para que o AutoCAD possa reconhecê-las como entidades e aplicar o tratamento adequado. Como exemplo, pode-se citar a sobrecarga das funções de manipulação de arquivos de dados. Esta manipulação permite a gravação dos dados destas entidades em arquivo e permitem também a implementação do comando UNDO (comando para desfazer uma ação).

Além das entidades “costumizadas”, existem outros tipos de armazenamento sem limite de tamanho para armazenar objetos. Estes objetos também devem sobrecarregar uma série de funções para possibilitar seu reconhecimento e tratamento pelo AutoCAD.

4.1.6 – Entidades, Objetos e o Banco de Dados do AutoCAD

Dentro do AutoCAD, existem dois tipos de elementos que podem ser criados quando se programa utilizando os recursos do ObjectARX. Na verdade, estes dois tipos de elementos são objetos sob o ponto de vista da linguagem C++. A diferença básica e determinante entre estes dois elementos é a
presença ou não de uma representação gráfica. Dessa forma, temos as
entidades, objetos com representação gráfica, e os objetos propriamente ditos.

No ObjectARX, existem duas classes disponíveis para derivação
específicas para cada tipo de objeto. Os objetos puros têm sua definição e
declaração feitas através da classe de nome AcDbObject. Já as entidades têm
sua definição e declaração através da classe AcDbEntity. As quatro primeiras
letras indicam que ambos os objetos são pertencentes ao banco de dados do
AutoCAD (AcDb – abreviação de AutoCAD Database). Na hierarquia de
classes, a classe AcDbEntity está abaixo de AcDbObject, ou seja, é derivada
de AcDbObject. Isto se deve ao fato de que as entidades são tipos de objetos
(AcDbObject) com a característica especial de possuir representação gráfica.

O banco de dados do AutoCAD contém uma estrutura funcional dividida
por objetos chamados de Container Objects. Estes objetos especiais têm a
função de armazenar entidades (AcDbEntity), objetos puros (AcDbObject) e
outros tipos de objetos. Esta estrutura exige uma série de procedimentos
seqüenciais para manipulação destes objetos. Devido à alocação de memória,
estes objetos devem ser manipulados, criados e excluídos segundo certos
cuidados e operações de segurança. Quando se cria objetos (entidades ou
objetos puros) estes devem ser corretamente armazenados no local apropriado
para recebê-los. Este procedimento é feito através da criação do objeto,
armazenamento de seus dados específicos e fechamento do objeto. O objetivo
do fechamento deste objeto é feito, por exemplo, para liberação de memória e
tratamento de notificações de eventos.

A estrutura do banco de dados é responsável pelo tratamento destes
objetos de acordo com o local onde estes são armazenados. Especificamente,
no caso de entidades e objetos puros, existem dois locais específicos onde
eles são armazenados (Figura 4.1.6-1).
Cada tabela é destinada a armazenar apenas o tipo de objeto descrito na estrutura acima (Figura 4.1.6-1). Para adicionar, remover ou editar cada componente destas tabelas existem procedimentos com uma execução sequencial de funções que efetuam todo o protocolo necessário para informar exatamente ao AutoCAD o tipo de operação que está sendo requisitada ou efetuada.

Os objetos são armazenados num Container Object chamado de Named Object Dictionary (NOD). O nome dicionário é bastante adequado e representa muito bem a estrutura deste objeto. O objetivo do NOD é guardar uma coleção de ponteiros para objetos puros e suas respectivas chaves. Cada objeto pode ser organizado dentro de outro dicionário. Isto é possível devido à estrutura de
árvore do NOD (Figura 4.1.6-2) que permite a criação e organização de vários dicionários dentro do NOD.

![Diagrama do NOD](image)

**Estrutura do NOD**

- **NOD**
  - **CONFIG**
  - **NUM**
  - **GRID**
  - **INC**

**Dicionário NOD contendo outros dois Dicionários (CONFIG e GRID) e dois Objetos (NUM e INC).**

- **NOD**
  - **CONFIG**
    - **CFG1**
    - **CFG2**
  - **NUM**
  - **GRID**
    - **GRD1**
    - **GRD2**
    - **SET1**
    - **SET2**
    - **GRD3**
  - **INC**

**Dicionário NOD expandido. Dicionário CONFIG contendo os objetos CFG1 e CFG2. Dicionário GRID contendo o dicionário GRD2 (contendo os objetos SET1 e SET2) e os objetos GRD1 e GRD3. NOD contendo ainda os objetos NUM e INC.**

Figura 4.1.6-2 – Estrutura do NOD (Named Object Dictionary).

As extremidades dos dicionários contêm chaves (Keys) associadas a ponteiros para objetos. Cada dicionário nada mais é do que uma coleção de ponteiros associados a nomes que identificam cada um destes dentro do dicionário. Quanto mais organizada for a estrutura do dicionário, mais fácil será o acesso aos objetos devido a busca imediata pelo dicionário de interesse.

As entidades são armazenadas numa tabela específica chamada Block Table. Dentro desta tabela, existem os registros chamados de Block Table Records onde existem as entradas onde são armazenadas as entidades. Estas entradas se chamam MODEL SPACE e PAPER SPACE (Figura 4.1.6-3). A entrada PAPER SPACE é destinada a receber entidades que serão utilizadas apenas no ambiente chamado de modelamento no papel dentro do AutoCAD. A entrada MODEL SPACE, a que será utilizada pelo Modelador, é destinada a
receber as entidades modeladas no espaço tridimensional no ambiente real do AutoCAD.

Outros tipos de objetos como Layers, LineType, Text Styles, etc., são armazenados em suas tabelas e têm características especiais. A cada novo desenho iniciado dentro do AutoCAD algumas destas tabelas são inicialmente preenchidas:

- **LineType Table**: Estilo de linha contínua;
- **Text Style**: Estilo de texto padrão;
- **Layer**: Camada de desenho 0 (cor branca/preta).
4.2 – Criação de Entidades Básicas

Inicialmente, serão implementadas, de forma isolada, as entidades estruturais Nó, Barra e Face. Esta criação isolada permite a determinação de dados iniciais que estão contidos nestas entidades de forma independente das demais. Além dos dados, também serão definidas funções particulares que determinam o comportamento isolado de cada entidade.

Após a implementação isolada de cada entidade, serão criadas funções para controlar o funcionamento em conjunto destas entidades de forma a criar dispositivos para o modelamento de uma estrutura contendo os três tipos de entidades estruturais. Este funcionamento em conjunto exige uma série de verificações de consistência de dados para prevenir e evitar operações ilegais e inadequadas do usuário.

Para o monitoramento das operações realizadas pelo usuário nas entidades estruturais, serão implementados reatores. Estes reatores permitem que qualquer mudança realizada em cada entidade seja monitorada e, a partir desta notificação, correções e ajustes sejam efetuados de acordo. Além dos reatores nas entidades, também serão implementados os reatores de comandos para fornecer notificações emitidas por comandos disparados dentro do AutoCAD, sejam eles nativos ou criados pelo próprio Modelador.
4.2.1 – Entidade Node (Nó)

O primeiro elemento representativo de uma estrutura é aquele que armazena dados sobre as conexões entre Barras, ou seja, o Nó. Este elemento, que é representado em termos de programação pela entidade **Node**, contém uma série de dados e funções que determinam seu comportamento estrutural, modificações aplicáveis e manipulação dentro do ambiente do AutoCAD.

Os dados contidos inicialmente no elemento Nó são suficientes para determinar o seu comportamento isolado. Estes dados são parte do objeto e estão encapsulados em uma classe chamada **Node**. Algumas destas funções são herdadas da classe **AcDbEntity**, classe base da biblioteca ObjectARX, de onde a classe **Node** é derivada. Estas funções podem ser redefinidas para tornar o comportamento desta entidade coerente com o comportamento objetivado pelo Modelador Estrutural. Dessa forma, as características desta entidade são determinadas pelos dados e funções contidas e encapsuladas na classe **Node** que pode ainda ser acrescida de dados e funções de acordo com a evolução do projeto.

Além das funções herdadas da classe superior, existem outras funções específicas da classe **Node** que cuidam da manipulação dos dados e do gerenciamento estrutural e funcional da entidade. Estas funções somente estão disponíveis nesta classe e podem ser acessadas através da utilização de ponteiros C++ de forma similar às funções sobrecarregadas.

A seguir, são descritas as características, dados e funções da entidade **Node** no contexto programático e estrutural.
4.2.1.1 – Variáveis Membros de Node

As variáveis membros da classe Node contêm, a princípio, informações sobre suas definições geométricas, conexões com Barras e identificação. A seguir, é apresentada a lista com as variáveis definidas nesta etapa do projeto (Figura 4.2.1.1-1).

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lps</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do ponto central de inserção da entidade Node.</td>
<td>Permanente</td>
</tr>
<tr>
<td>LpsText</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do ponto de inserção do texto de identificação da entidade Node.</td>
<td>Permanente</td>
</tr>
<tr>
<td>enormal</td>
<td>AcGeVector3d</td>
<td>Componentes do vetor normal que determinam o plano de desenho da entidade Node.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Nodeld</td>
<td>Adesk::Int16</td>
<td>Número contendo a identificação numérica do Nó.</td>
<td>Permanente</td>
</tr>
<tr>
<td>BarsList</td>
<td>ADbObjectIdArray</td>
<td>Lista de identificadores das barras conectadas ao Nó.</td>
<td>Permanente</td>
</tr>
<tr>
<td>nrelRot</td>
<td>Adesk::UInt8</td>
<td>Codificação para liberação do Nó à rotação.</td>
<td>Permanente</td>
</tr>
<tr>
<td>nrelTrans</td>
<td>Adesk::UInt8</td>
<td>Codificação para liberação do Nó à translação.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

Figura 4.2.1.1-1 – Lista de variáveis membros de Node.

4.2.1.2 – Funções Membros de Node

As funções membros da classe Node têm como objetivo determinar e controlar o funcionamento da entidade Node a nível estrutural e gráfico (no
ambiente do AutoCAD). A seguir, é apresentada a lista com as funções definidas nesta etapa do projeto (Figura 4.2.1.2-1).

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>worldDraw</td>
<td>Acad::Boolean</td>
<td>Criar a representação gráfica da entidade Node.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getOsnapPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar a localização dos pontos de precisão.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getGripPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar os pontos para habilitação de GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>moveGripPointsAt</td>
<td>Acad::ErrorStatus</td>
<td>Habilitar a movimentação dos pontos com GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>transformBy</td>
<td>Acad::ErrorStatus</td>
<td>Aplicar uma matriz de transformação tridimensional sobre a entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>GetTransformedCopy</td>
<td>Acad::ErrorStatus</td>
<td>Aplicar uma matriz de transformação durante a cópia da entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>list</td>
<td>void</td>
<td>Listar no AutoCAD os dados da entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>osnapCen</td>
<td>Acad::ErrorStatus</td>
<td>Atualizar os pontos da entidade quando a opção CENTER estiver habilitada.</td>
<td>Específica</td>
</tr>
<tr>
<td>setSp</td>
<td>void</td>
<td>Armazenar o ponto central de inserção da entidade Node.</td>
<td>Específica</td>
</tr>
<tr>
<td>getSp</td>
<td>AcGePoint3d</td>
<td>Retornar o ponto central de inserção da entidade Node.</td>
<td>Específica</td>
</tr>
<tr>
<td>setText</td>
<td>void</td>
<td>Armazenar o ponto de inserção do texto de identificação do Nó.</td>
<td>Específica</td>
</tr>
<tr>
<td>setNormal</td>
<td>void</td>
<td>Armazenar o vetor normal da entidade Node.</td>
<td>Específica</td>
</tr>
<tr>
<td>setNodeId</td>
<td>void</td>
<td>Armazenar o número de identificação do Nó.</td>
<td>Específica</td>
</tr>
<tr>
<td>getModeId</td>
<td>Adesk::Int16</td>
<td>Retornar o número de identificação do Nó.</td>
<td>Específica</td>
</tr>
<tr>
<td>------</td>
<td>--------------</td>
<td>------------------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>AppendBarsList</td>
<td>void</td>
<td>Adicionar a identificação de uma barra à lista de barras.</td>
<td>Específica</td>
</tr>
<tr>
<td>RemoveBar</td>
<td>void</td>
<td>Remover uma barra da lista de barras.</td>
<td>Específica</td>
</tr>
<tr>
<td>ContainsBar</td>
<td>Adesk::Boolean</td>
<td>Retornar verdadeiro se a barra está contida na lista de barras. Retornar falso em caso contrário.</td>
<td>Específica</td>
</tr>
<tr>
<td>EmptyListBar</td>
<td>Adesk::Boolean</td>
<td>Retornar verdadeiro se a lista de barras está vazia, caso contrário retornar falso.</td>
<td>Específica</td>
</tr>
<tr>
<td>GetBar</td>
<td>AcDbObjectId</td>
<td>Retornar a identificação da barra contida na lista de barras.</td>
<td>Específica</td>
</tr>
<tr>
<td>GetBarsListLen</td>
<td>int</td>
<td>Retornar o comprimento da lista de barras.</td>
<td>Específica</td>
</tr>
<tr>
<td>setTrans</td>
<td>void</td>
<td>Armazenar o código de liberação à translação.</td>
<td>Específica</td>
</tr>
<tr>
<td>getTrans</td>
<td>int</td>
<td>Retornar o código de liberação à translação.</td>
<td>Específica</td>
</tr>
<tr>
<td>setRot</td>
<td>void</td>
<td>Armazenar o código de liberação à rotação.</td>
<td>Específica</td>
</tr>
<tr>
<td>getRot</td>
<td>int</td>
<td>Retornar o código de liberação à rotação.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

Figura 4.2.1.2-1 – Lista de funções membros de Node.

### 4.2.1.3 – Representação Gráfica da Entidade Node

Como mencionado anteriormente, o Modelador trata as entidades no espaço tridimensional. A visualização do elemento estrutural Nó deve ser estabelecida de forma a permitir a percepção de sua existência de um ponto de visão arbitrário. Para proporcionar esta visualização foi estabelecido que a entidade **Node** seja composta por um grupo de três circunferências distribuídas
nos planos XY, XZ e YZ e ainda por um texto, no plano XY, representativo da identificação de cada Nó (Figura 4.2.1.3-1).

Figura 4.2.1.3-1 – Representação gráfica da entidade Node.

Esta representação permite uma visualização do Nó em qualquer plano e em um ponto arbitrário de visão em perspectiva. No caso do texto, a visualização ficaria confusa se utilizada nos três planos, sendo assim, foi escolhido o plano XY como o mais frequente para ser utilizado como plano de desenho do texto.

A entidade Node se subdivide em 2 subentidades. A primeira formada pelas três circunferências e a segunda pelo texto representativo. Esta subdivisão permite que cada subentidade seja criada em uma camada de desenho diferente, ou seja, em Layer diferente. Cada uma das subentidades tem sua posição determinada por um ponto de referência. Para as circunferências, tem-se o ponto armazenado pela variável Ips e para o texto representativo, tem-se o ponto armazenado em IpsText.

O posicionamento do texto representativo pode ser comprometido caso ocorra uma sobreposição de entidades. Para possibilitar a movimentação da subentidade texto representativo e para possibilitar também, numa etapa posterior, a movimentação do próprio Nó, foram implementados pontos de GRIP. Estes pontos permitem que, através de uma seleção dentro do AutoCAD, as entidades sejam movidas, rotacionadas, esticadas, etc. Este
dispositivo torna a manipulação das entidades mais fácil e intuitiva (Figura 4.2.1.3-2).

**Figura 4.2.1.3-2 – Localização dos pontos de GRIP na entidade Node.**

Quando a entidade **Node** é selecionada no AutoCAD estes pontos são habilitados e as coordenadas das variáveis **lps** e **lpsText** são atualizadas de acordo com a transformação espacial aplicada em cada ponto de GRIP. Esta nova posição é armazenada pela entidade **Node** nas variáveis específicas quando o desenho é gravado. A variável **lpsText**, que determina a posição do texto, é independente de **lps** além de ser permanente e membro da classe **Node**.

### 4.2.1.4 – Lista de Barras

As entidades representativas do elemento Nó de uma estrutura devem conter informações sobre as barras que incidem em cada Nó. Dentro do AutoCAD, cada entidade adicionada ao banco de dados tem uma identificação única, uma espécie de nome. Esta identificação, representada no ObjectARX pela classe **AcDbObjectId**, é permanente e permanece coerente com as informações contidas no arquivo DWG. A codificação é um número hexadecimal gerado automaticamente pelo AutoCAD. As traduções são feitas automaticamente pelo AutoCAD, não sendo necessária qualquer operação adicional pelo programador.
Estas identificações são armazenadas em cada entidade **Node** em forma de *array* dinâmico do tipo **AcDbObjectIdArray**, através da variável membro **BarsList** (Figura 4.2.1.4-1). A utilização deste tipo de vetor de armazenamento é plenamente satisfatória devido à sua estrutura de tamanho dinâmico, ou seja, este vetor tem o comprimento exatamente igual ao número de barras conectadas a cada Nó.

A lista de barras permite ainda a fácil inserção ou remoção de barras quando necessário (funções **AppendBarsList** e **RemoveBar**). Esta informação tem também o objetivo de fornecer dados para a manipulação conjunta das entidades **Node** e **Bar** como será visto em breve. Através de funções específicas do ObjectARX é possível extrair informações das entidades a partir de sua identificação. Dessa forma, quando a entidade **Node** estiver sendo acessada, pode-se obter informações de qualquer barra que esteja contida na lista de barras da entidade **Node**. A manipulação da lista de barras é feita através de funções membros definidas na classe descritas no item 4.2.1.2.
4.2.1.5 – Verificações de Consistência

Cada entidade Node criada deve ser única em relação às suas coordenadas tridimensionais, ou seja, em cada ponto deve existir somente um Nó. Quando o usuário especifica um ponto onde será criada uma entidade Node é fornecido um ponto tridimensional como parâmetro de entrada para a função de verificação. Este parâmetro é então comparado às entidades do tipo Node contidas no banco de dados do AutoCAD. Para efetuar esta verificação foi criada um função utilitária chamada verifyNode (Figura 4.2.1.5-1).

![Fluxograma da função verifyNode](image_url)

Assim, cada vez que a função de criação de entidades do tipo Node é chamada, a função verifyNode deve ser chamada antes para que as verificações de banco de dados sejam corretamente averiguadas e para que não sejam criadas entidades Node duplicadas, ou seja, com as mesmas coordenadas espaciais de algum Nó já existente.
A estrutura desta função é bastante flexível possibilitando assim sua utilização em outras partes do programa. A verificação é feita de uma só vez, abrindo-se todas as entidades contidas no banco de dados com o filtro de verificação do tipo de entidade. Dessa forma somente serão verificadas as entidades do tipo Node sendo as demais ignoradas pela função.

O objetivo do retorno da identificação da entidade Node encontrada, caso já exista algum Node com as coordenadas passadas para a função, é fornecer parâmetros que possam proporcionar a criação de uma Barra partindo de um Nó já existente. Esta identificação retornada pela função será passada à entidade Bar e as informações desta Barra serão armazenadas na entidade Node já existente. Este mecanismo permite a manutenção das incidências de Barras e Nós de forma coerente.

4.2.1.6 – Restrição Nodal

Cada entidade Node contém também informações sobre seu comportamento estrutural quanto às liberações à rotação e à translação. Estas informações estão contidas nas variáveis membros nrelRot e nrelTrans respectivamente e são acessíveis através das funções membros getRot, setRot, getTrans e setTrans. Como existem algumas combinações possíveis entre os tipos de liberações, foram estabelecidas codificações (Figura 4.2.1.6-1) para que se pudesse armazenar apenas um número inteiro contendo todas as informações necessárias para codificar rotação e translação.
Dessa forma, pode-se otimizar o armazenamento de variáveis na entidade **Node**. Cada entidade **Node** criada recebe, por padrão, as codificações 8 (oito) para liberação à rotação e 8 (oito) para liberação à translação. Esta codificação fica, a princípio, congelada e não aparece na tela de desenho. Através da função de controle de visualização, descrita no item 4.5 deste projeto, é possível o controle da exibição ou não das restrições. Para alterar o valor das restrições nodais foi criado um comando específico do Modelador. Utilizando o comando **NREL** pode-se acessar as entidades **Node** e alterar o valor da codificação de suas restrições.

Ao se executar o comando **NREL**, são disponibilizadas opções de edição para o usuário. O comando é executado através de um quadro de diálogo permitindo também uma visualização de exemplos da representação gráfica da entidade **Node** com as restrições habilitadas ou não (Figura 4.2.1.6-2).
Esta codificação é então interpretada durante a exportação dos dados para determinar o comportamento estrutural de cada Nó da estrutura. Da mesma forma como foi realizada a representação gráfica do próprio Nó, a representação das restrições é feita através de desenhos nos três planos tridimensionais principais.

Esta distribuição da representação em planos principais permite que o usuário seja capaz de visualizar a disposição das restrições de forma independente em cada plano. Para representar a restrição quanto à rotação foram escolhidas as figuras círculo e quadrado onde o lado do quadrado é igual ao diâmetro do círculo. Esta representação permite que se tivermos, por exemplo, uma restrição à rotação em um plano e uma liberação em outro estes desenhos não influam um sobre o outro. Ao ser visto de lado, o quadrado, visto como uma linha, se sobrepõe à linha do círculo e vice-versa.
Para representar as liberações quanto à translação foi escolhida como figura uma reta duplicada. A presença desta reta indica liberação e sua ausência, restrição (Figura 4.2.1.6-3).

![Diagrama de liberações e restrições](image)

Figura 4.2.1.6-3 – Exemplo de representação de liberações e restrições.

### 4.2.2 – Entidade Bar (Barra)

A segunda entidade representativa de uma estrutura, a Barra, armazena dados sobre as conexões entre Nós. A entidade Bar, que representa em termos de programação o elemento Barra da análise estrutural, contém uma série de dados e funções que determinam seu comportamento estrutural, modificações aplicáveis e manipulação dentro do ambiente do AutoCAD.

Os dados contidos inicialmente na entidade Bar determinam seu comportamento isolado. Estes dados pertencem ao elemento Barra e estão
encapsulados na classe Bar. Algumas destas funções são herdadas da classe AcDbCurve, classe derivada de AcDbEntity, que permite a habilitação do comando OFFSET (comando nativo do AutoCAD). Estas funções podem ser redefinidas para tornar o comportamento desta entidade coerente com o comportamento objetivado pelo Modelador Estrutural. Dessa forma, as características desta entidade são determinadas pelos dados e funções contidas e encapsuladas na classe Bar que pode ainda ser acrescida de dados e funções de acordo com a evolução do projeto.

Além das funções herdadas da classe superior existem outras funções específicas da classe Bar que cuidam da manipulação dos dados e do gerenciamento estrutural e funcional da entidade. Estas funções somente estão disponíveis nesta classe e podem ser acessadas através da utilização de ponteiros C++ de forma similar às funções sobrecarregadas.

A seguir, são descritas as características, dados e funções da entidade Bar no contexto programático e estrutural.

4.2.2.1 – Variáveis Membros de Bar

As variáveis membros da classe Bar contêm, a princípio, informações sobre suas definições geométricas, Nó inicial e final e identificação. A seguir, é apresentada a lista com as variáveis definidas nesta etapa do projeto (Figura 4.2.2.1-1).

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>lps</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do ponto inicial da Barra.</td>
<td>Permanente</td>
</tr>
<tr>
<td>lpe</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do ponto final da Barra.</td>
<td>Permanente</td>
</tr>
<tr>
<td>StartNode</td>
<td>AcDbObjectId</td>
<td>Identificação do Nó inicial da Barra.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Nome</td>
<td>Retorno</td>
<td>Objetivos</td>
<td>Tipo</td>
</tr>
<tr>
<td>--------------------</td>
<td>----------------------</td>
<td>--------------------------------------------------------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>worldDraw</td>
<td>Adesk::Boolean</td>
<td>Criar a representação gráfica da entidade Bar.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getOsnapPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar a localização dos pontos de precisão.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getGripPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar os pontos para habilitação de GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getStretchPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar os pontos para alterar o comprimento da Barra.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>moveStretchPointsAt</td>
<td>Acad::ErrorStatus</td>
<td>Habilitar a movimentação dos pontos com GRIP para alterar o comprimento da Barra.</td>
<td>Sobrecarregada</td>
</tr>
</tbody>
</table>

Figura 4.2.2.1-1 – Lista de variáveis membros de Bar.

4.2.2.2 – Funções Membros de Bar

As funções membros da classe Bar têm como objetivo determinar e controlar o funcionamento da entidade Bar a nível estrutural e gráfico (no ambiente do AutoCAD). A seguir, é apresentada a lista com as funções definidas nesta etapa do projeto (Figura 4.2.2.2-1).
<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Description</th>
<th>Overload</th>
</tr>
</thead>
<tbody>
<tr>
<td>moveGripPointsAt</td>
<td>Acad::ErrorStatus</td>
<td>Habilitar a movimentação dos pontos com GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>transformBy</td>
<td>Acad::ErrorStatus</td>
<td>Aplicar uma matriz de transformação tridimensional sobre a entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getTransformedCopy</td>
<td>Acad::ErrorStatus</td>
<td>Efetivar uma matriz de transformação durante a cópia da entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>deepClone</td>
<td>Acad::ErrorStatus</td>
<td>Monitorar as operações de cópia sobre Barras e efetuar operações de verificação e ajuste de dados.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>explode</td>
<td>Acad::ErrorStatus</td>
<td>Permitir que uma linha imaginária seja criada a partir da Barra para proporcionar o comando OFFSET.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>list</td>
<td>void</td>
<td>Listar no AutoCAD os dados da entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>osnapEnd</td>
<td>Acad::ErrorStatus</td>
<td>Atualizar os pontos extremos da Barra quando a opção ENDPOINT estiver habilitada.</td>
<td>Específica</td>
</tr>
<tr>
<td>osnapMid</td>
<td>Acad::ErrorStatus</td>
<td>Atualizar o ponto médio da Barra quando a opção MIDPOINT estiver habilitada.</td>
<td>Específica</td>
</tr>
<tr>
<td>osnapPerp</td>
<td>Acad::ErrorStatus</td>
<td>Atualizar pontos da Barra quando a opção PERPENDICULAR estiver habilitada.</td>
<td>Específica</td>
</tr>
<tr>
<td>setSp</td>
<td>void</td>
<td>Armazenar o ponto inicial da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getSp</td>
<td>AcGePoint3d</td>
<td>Retornar o ponto inicial da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>setEp</td>
<td>void</td>
<td>Armazenar o ponto final da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getEp</td>
<td>AcGePoint3d</td>
<td>Retornar o ponto final da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>setStartNode</td>
<td>void</td>
<td>Armazenar a identificação do Nó inicial da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getStartNode</td>
<td>AcDbObjectId</td>
<td>Retornar a identificação do Nó inicial da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>setEndNode</td>
<td>void</td>
<td>Armazenar a identificação do Nó final da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getEndNode</td>
<td>AcDbObjectId</td>
<td>Retornar a identificação do Nó final da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>textFit</td>
<td>void</td>
<td>Ajustar o posicionamento do texto de acordo com a inclinação da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>setBarId</td>
<td>void</td>
<td>Armazenar o número de identificação da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getBarId</td>
<td>Adesk::Int16</td>
<td>Retornar o número de identificação da Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>setTpBar</td>
<td>void</td>
<td>Armazenar o código de identificação do tipo de Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>getTpBar</td>
<td>Adesk::Uint8</td>
<td>Retornar o código de identificação do tipo de Barra.</td>
<td>Específica</td>
</tr>
<tr>
<td>isPlanar</td>
<td>Adesk::Boolean</td>
<td>Retornar verdadeiro se a entidade é planar.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getPlane</td>
<td>Acad::ErrorStatus</td>
<td>Retornar o flag AcDb::kLinear.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getStartParam</td>
<td>Acad::ErrorStatus</td>
<td>Retornar o parâmetro inicial da curva.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getPointAtParam</td>
<td>Acad::ErrorStatus</td>
<td>Retornar um ponto a partir do parâmetro da curva.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getFirstDeriv</td>
<td>Acad::ErrorStatus</td>
<td>Retornar a primeira derivada da curva.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getClosestPointTo</td>
<td>Acad::ErrorStatus</td>
<td>Criar uma nova entidade a partir da Barra original a uma distância especificada pelo vetor passado no comando OFFSET do AutoCAD.</td>
<td>Sobregravada</td>
</tr>
<tr>
<td>getOffsetCurves</td>
<td>Acad::ErrorStatus</td>
<td>Criar uma nova entidade a partir da Barra original a uma distância especificada pelo vetor passado no comando OFFSET do AutoCAD.</td>
<td>Sobregravada</td>
</tr>
</tbody>
</table>

Figura 4.2.1.2-1 – Lista de funções membros de Bar.
4.2.2.3 – Representação Gráfica da Entidade Bar

A representação gráfica da entidade Bar é composta por uma linha tridimensional ligando duas entidades Node e um texto representativo com o número de identificação da Barra. Pode-se observar, a seguir, a representação de uma Barra arbitrária no plano XY ligando dois Nós (Figura 4.2.2.3-1).

Os números de identificação 5 e 14 são os Nós inicial e final, respectivamente, da entidade Bar. O texto 12 é o texto representativo da Barra.

Assim como a entidade Node, o texto de identificação da Barra está somente desenhado no plano XY. No item 4.2.4 será exemplificada a criação conjunta das entidades e o armazenamento das informações de incidência da Barra e seus respectivos Nós inicial e final. A entidade Bar se subdivide em 2 subentidades. A primeira formada pela linha que une os Nós inicial e final e a segunda pelo texto representativo. Esta subdivisão permite que cada subentidade seja criada em uma camada de desenho diferente. Cada uma das subentidades tem sua posição determinada por pontos de referência. Para a linha, tem-se os pontos armazenados pelas variáveis \( lps \) e \( lpe \) e, para o texto representativo, tem-se o ponto armazenado em \( lptext \).

O posicionamento do texto representativo é determinado pela inclinação da Barra e pelo ponto médio da linha que une os Nós inicial e final. A função membro \texttt{textFit} efetua o cálculo da inclinação, do vetor perpendicular à linha
da Barra e ainda centraliza o texto em função do número de dígitos. Além disso, na função de desenho worldDraw ocorre uma verificação da direção da Barra para que o texto seja desenhado na orientação correta. Esta orientação é determinada pelo posicionamento da Barra no plano XY conforme esquema apresentado a seguir (Figura 4.2.2.3-2).

![Diagrama de posicionamento e orientação do texto de identificação da Barra.](image)

Figura 4.2.2.3-2 – Posicionamento e orientação do texto de identificação da Barra.

Este posicionamento automático pode ser comprometido caso ocorra uma sobreposição de entidades. Para possibilitar a movimentação da subentidade texto representativo e para possibilitar também a alteração no comprimento da Barra foram implementados pontos de GRIP. Estes pontos permitem que, através de uma seleção dentro do AutoCAD, as entidades sejam movidas, rotacionadas, esticadas, encurtadas, etc. Dessa forma, a manipulação das entidades é realizada facilmente (Figura 4.2.2.3-3).
Figura 4.2.1.3-3 – Localização dos pontos de GRIP na entidade Bar.

Quando a entidade **Bar** é selecionada no AutoCAD, estes pontos são habilitados e as coordenadas das variáveis **lps**, **lpe** e **lptext** são atualizadas de acordo com a transformação espacial aplicada em cada ponto de GRIP. Esta nova posição é armazenada pela entidade **Bar** quando o desenho é gravado. A variável **lptext**, que determina a posição do texto, depende de **lps**, **lpe**, da orientação da Barra e do número de dígitos do texto.

**4.2.2.4 – Identificação de Nós e de Pontos Final e Inicial**

Nas Barras, além do armazenamento das coordenadas dos pontos inicial e final, é necessário também o armazenamento das identificações dos Nós inicial e final. Da mesma forma, como foi implementada a lista de Barras na entidade **Node**, na entidade **Bar**, estas identificações servirão como parâmetros para as verificações de coerência.
Através das variáveis **StartNode** e **EndNode**, a identificação, ou seja, o **AcDbObjectId** dos Nós, é armazenada. Dependendo da forma como a Barra foi criada não necessariamente as coordenadas do Nó inicial e do ponto inicial da Barra serão iguais. A mesma observação vale para o Nó final (Figura 4.2.2.4-1).

![Diagrama de Barra e Nós](image)

**Figura 4.2.2.4-1 – Identificação de Nós e pontos da Barra.**

### 4.2.2.5 – Verificações de Consistência

Cada entidade **Bar** criada deve ser única em relação às suas coordenadas tridimensionais independente da ordem de criação dos seus Nós inicial e final. Quando uma entidade **Bar** é criada, são fornecidos dois pontos tridimensionais como parâmetros para a função de verificação.

Para efetuar esta verificação foi criada um função utilitária chamada **verifyBar**. Esta função é executada de acordo com o seguinte fluxograma (Figura 4.2.2.5-1):
Figura 4.2.2.5-1 – Fluxograma da função `verifyBar`.

Estes parâmetros são comparados às entidades do tipo `Bar` contidas no banco de dados do AutoCAD, levando-se em conta a possibilidade de Barras com Nós de mesma coordenadas invertidas (Nó inicial da Barra que está sendo criada igual ao Nó final de uma Barra existente e vice-versa). Caso exista uma Barra idêntica ou invertida em relação à que está se pretendendo criar, a rotina de verificação retorna um código que evita a criação de uma Barra duplicada.

Dessa forma, cada vez que a função de criação de Barras é chamada, a função `verifyBar` é chamada antes para que as verificações de banco de dados sejam corretamente verificadas e para que não sejam criadas entidades
Bar duplicadas, ou seja, com as mesmas coordenadas espaciais invertidas ou não.

A estrutura desta função é bastante flexível possibilitando assim sua utilização em outras partes do programa. A verificação é feita de uma só vez, abrindo-se todas as entidades contidas no banco de dados com o filtro de verificação do tipo de entidade. Dessa forma, somente serão verificadas as entidades do tipo Bar sendo as demais ignoradas pela função.

O objetivo do retorno da identificação da entidade Bar encontrada, caso já exista alguma Barra com as mesmas coordenadas (invertidas ou não) passadas para a função, é fornecer parâmetros que possam proporcionar a criação de uma Barra partindo de um Nó existente como visto no item 4.2.1.5.

4.2.3 – Comportamento Conjunto de Nó e Barra

A partir das duas entidades básicas criadas, Node e Bar, pode-se estabelecer e implementar a rotina de criação de Barras e Nós de forma conjunta. Para cada Barra criada, serão criados ou conectados Nós correspondentes. Esta sistematização permite a criação de uma estrutura tridimensional baseada em Nós e Barras. Não existe, a princípio, sentido na criação de Nós isolados, ou seja, sem Barras conectadas. Assim, a criação da estrutura unifilar será feita através da entrada de pontos que determinarão a criação de Barras.

A estrutura da rotina de criação de Barras e Nós é sistematizada de forma similar ao comando de criação de linhas simples no AutoCAD mantendo assim o funcionamento do Modelador semelhante ao do próprio AutoCAD. A criação sucessiva de Barras é feita através do comando BAR do Modelador.
Este comando pedirá ao usuário que entre com pontos espaciais especificados por coordenadas ou determinados na tela gráfica do AutoCAD.

A precisão de trabalho do AutoCAD permite que sejam especificados pontos de precisão na tela gráfica. Assim, para que se determine exatamente o ponto central de um Nó (coincidente com o ponto final ou inicial de uma Barra), o usuário deve habilitar a opção de Object Snap (OSNAP) do tipo CENTER (ponto central). Nos itens 4.2.1 e 4.2.2, foram especificados os pontos de Object Snap para as entidades Node e Bar. Para a entidade Node, foi especificado apenas o ponto central do Nó quando a opção CENTER do AutoCAD estiver habilitada. Para a entidade Bar, foram especificados os pontos de extremidade da Barra como ENDPOINT e o ponto médio da Barra como MIDPOINT. O uso da precisão garante que as coordenadas de criação de uma Barra e de um Nó sejam exatamente as especificadas pelo usuário de forma numérica ou visual.

Durante a criação de Barras e, consequentemente, de Nós, são efetivadas verificações utilizando as funções descritas nos itens 4.2.1 e 4.2.2, verifyNode e verifyBar. Além destas duas funções uma terceira função, chamada AppendNodeBar (Figura 4.2.3-1), foi criada para tratar de casos especiais de criação de Barras como: Barras partindo de um Nó até um ponto qualquer, Barras partindo de um ponto qualquer até um Nó e Barras partindo de um Nó até outro Nó. Esta função recebe três identificações (AcDbObjectId) sendo duas de Nós e a terceira de uma Barra. A função AppendNodeBar verifica e atualiza os dados sobre conectividade entre Nós (lista de Barras) e Barras (Nós inicial e final).
Além dessas verificações, foi criada ainda uma rotina de verificação da ordem de entrada dos pontos inicial e final da Barra. O critério adotado para esta ordem de entrada obedece a orientação das Barras conforme a Figura 4.2.2.3-2 do item 4.2.2. Se o usuário efetuar a entrada inversa dos pontos, de acordo com o padrão estabelecido, a rotina ajusta automaticamente a ordem dos pontos para criar a Barra na orientação correta. Esta coerência, além de tornar mais lógica a organização de Nós, permite que os Nós iniciais, utilizados como referência em outras rotinas, permaneçam compatíveis com a
distribuição geométrica da Barra. A rotina de criação de Barras está descrita no fluxograma da Figura 4.2.3-2.

![Fluxograma de criação de Barras](image)

Figura 4.2.3-2 – Fluxograma da rotina de criação de Barras.

A estrutura do fluxograma da Figura 4.2.3-2 cobre as três possibilidades de criação de Barras e evita a criação de entidades duplicadas (Figura 4.2.3-3). O laço de repetição do fluxograma representa a sequência de criação de uma
Barra permitindo que, ao final, se outro ponto for fornecido pelo usuário, o laço seja repetido partindo do último ponto fornecido ou invertido. Esta estruturação permite que diversas Barras sejam criadas de forma sequencial, mesmo que sejam escolhidos Nós já existentes ou Barras já existentes. As rotinas de verificação efetuam as operações necessárias para manter a correta conectividade da estrutura criada.

Figura 4.2.3-3 – Tipos de criação de Barras.

4.2.4 – Entidade Face (Superfície)

Para complementar a definição dos elementos estruturais foi criada a entidade **Face**. Esta entidade, que representa em termos de programação o elemento Superfície da análise estrutural, contém uma série de dados e funções que determinam seu comportamento estrutural, modificações aplicáveis e manipulação dentro do ambiente do AutoCAD.
Na entidade **Face** estão contidos, inicialmente, dados que determinam seu comportamento isolado. Estes dados do elemento Superfície estão encapsulados na classe **Face**. Algumas destas funções são herdadas da classe superior **AcDbEntity**. Estas funções podem ser redefinidas para tornar o comportamento desta entidade coerente com o comportamento objetivado pelo Modelador Estrutural. Assim, as características da entidade **Face** são determinadas pelos dados e funções contidas e encapsuladas na classe **Face** que pode ainda ser acrescida de dados e funções de acordo com a evolução do projeto.

Além das funções herdadas da classe superior existem outras funções específicas da classe **Face** que cuidam da manipulação dos dados e do gerenciamento estrutural e funcional da entidade. Estas funções somente estão disponíveis nesta classe e podem ser acessadas através da utilização de ponteiros C++ de forma similar às funções sobrecarregadas.

A seguir, são descritas as características, dados e funções da entidade **Face** no contexto programático e estrutural.
4.2.4.1 – Variáveis Membros de Face

As variáveis membros da classe Face contêm, a princípio, informações sobre suas definições geométricas, lados e identificação. A seguir, é apresentada a lista com as variáveis definidas nesta etapa do projeto (Figura 4.2.4.1-1).

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>lps</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do centro de gravidade da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>lpsText</td>
<td>AcGePoint3d</td>
<td>Coordenadas tridimensionais do ponto de inserção do texto de identificação da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>enormal</td>
<td>AcGeVector3d</td>
<td>Vetor normal ao plano da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Faceld</td>
<td>Adesk::Int16</td>
<td>Número de identificação da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>vertexList</td>
<td>AcGePoint3dArray</td>
<td>Lista dos pontos tridimensionais de cada vértice da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Repre</td>
<td>Adesk::Int16</td>
<td>Codificação do tipo de representação gráfica da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>ang</td>
<td>double</td>
<td>Ângulo de trabalho da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>pti</td>
<td>AcGePoint3d</td>
<td>Ponto inicial do eixo local da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>ptf</td>
<td>AcGePoint3d</td>
<td>Ponto final do eixo local da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>ptaux</td>
<td>AcGePoint3d</td>
<td>Ponto auxiliar.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Control</td>
<td>Adesk::Int16</td>
<td>Codificação de identificação da geometria da Superfície.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Per</td>
<td>double</td>
<td>Porcentagem de desenho da direção de trabalho da Superfície.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

Figura 4.2.4.1-1 – Lista de variáveis membros de Face.
4.2.4.2 – Funções Membros de Face

As funções membros da classe *Face* têm como objetivo determinar e controlar o funcionamento da entidade *Face* a nível estrutural e gráfico (no ambiente do AutoCAD). A seguir, é apresentada a lista com as funções definidas nesta etapa do projeto (Figura 4.2.4.2-1).

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>worldDraw</td>
<td>Adesk::Boolean</td>
<td>Criar a representação gráfica da entidade Face.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getGripPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar os pontos para habilitação de GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>getStretchPoints</td>
<td>Acad::ErrorStatus</td>
<td>Determinar os pontos para alterar a geometria da Superfície.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>moveStretchPointsAt</td>
<td>Acad::ErrorStatus</td>
<td>Habilitar a movimentação dos pontos com GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>moveGripPointsAt</td>
<td>Acad::ErrorStatus</td>
<td>Habilitar a movimentação dos pontos com GRIP.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>transformBy</td>
<td>Acad::ErrorStatus</td>
<td>Aplicar uma matriz de transformação tridimensional sobre a entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>deepClone</td>
<td>Acad::ErrorStatus</td>
<td>Monitorar as operações de cópia sobre Barras e efetuar operações de verificação e ajuste de dados.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>list</td>
<td>void</td>
<td>Listar no AutoCAD os dados da entidade.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>setSp</td>
<td>AcGePoint3d</td>
<td>Armazenar o centro de gravidade da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setText</td>
<td>void</td>
<td>Armazenar o número de identificação da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>Function</td>
<td>Return Type</td>
<td>Description</td>
<td>Category</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>setNormal</td>
<td>void</td>
<td>Armazenar o vetor normal da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>textFit</td>
<td>void</td>
<td>Ajustar o posicionamento do texto de acordo com a direção de trabalho da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setFaceld</td>
<td>void</td>
<td>Armazenar o número de identificação da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getFaceld</td>
<td>Adesk::Int16</td>
<td>Retornar o número de identificação da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>appendVertex</td>
<td>void</td>
<td>Armazenar vértices na Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>nullArray</td>
<td>void</td>
<td>Zerar a lista de vértices.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setCG</td>
<td>void</td>
<td>Armazenar o centro de gravidade.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getCG</td>
<td>AcGePoint3d</td>
<td>Retornar o centro de gravidade.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setFaceType</td>
<td>void</td>
<td>Determinar o tipo de Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setRepre</td>
<td>void</td>
<td>Armazenar o tipo de representação da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getRepre</td>
<td>Adesk::Int16</td>
<td>Retornar o tipo de representação da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>setAng</td>
<td>void</td>
<td>Armazenar o ângulo de trabalho da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getAng</td>
<td>double</td>
<td>Retornar o ângulo de trabalho da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getVlen</td>
<td>int</td>
<td>Retornar o número de vértices da Superfície.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getVertex</td>
<td>AcGePoint3d</td>
<td>Retornar o ponto tridimensional de cada vértice.</td>
<td>Especifica</td>
</tr>
<tr>
<td>getpti</td>
<td>AcGePoint3d</td>
<td>Retornar o ponto inicial do eixo local da Superfície.</td>
<td>Especifica</td>
</tr>
</tbody>
</table>

Figura 4.2.4.2-1 – Lista de funções membros de Face.
4.2.4.3 – Representação Gráfica da Entidade Face

A representação gráfica da entidade Face é composta por um polígono de três ou mais lados e um texto representativo com o número de identificação da Superfície. Pode-se observar, a seguir, a representação de uma Face arbitrária no plano XY (Figura 4.2.4.3-1).

![Diagrama de Face](image)

Figura 4.2.4.3-1 – Representação gráfica da entidade Face.

O texto de identificação da Superfície está somente desenhado no plano XY. A entidade Face se subdivide em 3 subentidades. A primeira formada pelo polígono do contorno da superfície, a segunda pelo texto representativo e a terceira pela identificação da direção de trabalho da Superfície. Maiores detalhes sobre a representação geométrica da entidade Face podem ser vistos em [Hüt98].

4.2.4.4 – Verificações de Consistência

Cada entidade Face deve ser única em relação ao seus lados. Quando uma entidade Face é criada, são fornecidos pontos no espaço, determinando os lados da Superfície (contorno) que são utilizados como parâmetros para a função de verificação.
Para efetuar esta verificação, foi criada um função utilitária chamada **verifyFace**. Esta função é executada de acordo com o seguinte fluxograma (Figura 4.2.4.4-1):

![Fluxograma](image-url)

Figura 4.2.4.4-1 – Fluxograma da função **verifyFace**.

Estes parâmetros são comparados às entidades do tipo **Face** contidas no banco de dados do AutoCAD levando-se em conta a possibilidade de
Superfícies com geometria igual mas número de vértices diferentes serem consideradas iguais. Caso exista uma Superfície idêntica ou equivalente em relação à que está sendo criada, a rotina de verificação retorna um código que evita a criação de uma Superfície duplicada.

Assim, cada vez que a função de criação de Superfície é chamada, a função **verifyFace** é chamada antes para que as verificações de banco de dados sejam corretamente executadas e para que não sejam criadas entidades **Face** duplicadas.

A estrutura desta função é bastante flexível possibilitando assim sua utilização em outras partes do programa. A verificação é feita de uma só vez, abrindo-se todas as entidades contidas no banco de dados com o filtro de verificação do tipo de entidade. Dessa forma, somente serão verificadas as entidades do tipo **Face** sendo as demais ignoradas pela função.

### 4.2.4.5 – Criação Individual de Superfícies

A criação individual de Superfícies pode ser efetuada através do comando **FACE** do Modelador. O comando requisita a especificação de um ponto na tela gráfica do AutoCAD, a partir do qual será detectada e criada a Superfície. Para a criação da entidade **Face**, foi criada uma função que, a partir de um ponto passado como parâmetro, cria uma Superfície automaticamente. Esta criação é feita com o auxílio de um comando já existente no AutoCAD, o comando **BOUNDARY**. Este comando contém um algoritmo que detecta a fronteira mais próxima a partir de um ponto. Em cada trecho desta fronteira detectada (polígono) é criado um trecho de uma **POLYLINE** (entidade do AutoCAD que representa uma seqüência conectada de linhas).

Após esta detecção, tem-se uma entidade **POLYLINE** contendo todos os vértices que formam a linha de contorno da Superfície. A partir desta lista de
pontos, pode-se criar a entidade Face individualmente (Figura 4.2.4.5-1). Os vértices contidos na POLYLINE são então copiados para a nova entidade Face. Após a criação da Superfície, a entidade POLYLINE é automaticamente apagada.

Figura 4.2.4.5-1 – Fluxograma da função createOneFace.

Maiores informações sobre a metodologia para criação das Superfícies podem ser encontradas em [Hüt98].
4.2.4.6 – Criação Automática de Superfícies

A criação individual de Superfícies é trabalhosa e demorada quando se trata de uma estrutura contendo um grande número de Barras. Para otimizar e acelerar o processo de criação de Superfícies, foi criado o comando **FACES** no Modelador. Este comando faz uma detecção automática dos pontos de possível existência e cria, nestas regiões, Superfícies.

Para detectar os pontos de possível existência de Superfícies, foi criada uma rotina (**createFaces**) que busca, no banco de dados do AutoCAD, os pontos de existência dos Nós, aplica um pequeno deslocamento (nos eixos X e Y) a 45 graus a partir destes pontos, e chama o comando de criação individual de Superfícies (**createOneFace**) passando este novo ponto como parâmetro. Para acelerar mais ainda o processo de criação, as camadas de desenho (**Layers**) da entidade **Node** e a numeração da entidade **Bar** são congeladas (Figura 4.2.4.6-1).

![Fluxograma da função createFaces.](image)
4.3 – Implementações Avançadas

Após a definição e a implementação dos três elementos estruturais básicos, foram necessárias algumas implementações adicionais para controle, edição, cópia, numeração e renumerar destas entidades.

A estrutura das três classes é bastante semelhante no que diz respeito às suas funções herdadas das classes fornecidas pelo ObjectARX. Tirando proveito desta semelhança, a implementação de novas funções pode ser feita, em alguns casos, de forma genérica, para atender aos três tipos de entidades.

O objetivo destas funções é facilitar e tornar o mais amigável possível a interface do Modelador. Além disso, para que se torne possível a utilização de comandos nativos do AutoCAD, foram implementadas funções de controle de comandos. Estas funções têm como objetivo aplicar ajustes e correções automáticas quando algum comando nativo do AutoCAD é executado afetando as entidades do Modelador.

A seguir, são descritas as implementações de funções e procedimentos sobre as entidades de forma conjunta e isolada. A sincronia e coerência entre as modificações realizadas nas entidades por estas funções garante o funcionamento correto do Modelador e a limitação dos erros que, possivelmente, poderiam ser cometidos pelo usuário.
4.3.1 – Operações no Banco de Dados do AutoCAD

Para facilitar a adição das entidades criadas ao banco de dados do AutoCAD, foi criada a função `append`. Esta função efetua o procedimento padrão do ObjectARX através de uma sequência de chamadas a funções utilitárias.

Cada entidade criada dentro do ambiente ObjectARX deve utilizar o operador `NEW` da linguagem C++. Com a utilização deste operador tem-se disponível um ponteiro para este novo objeto. Com o ponteiro, pode-se acessar as funções e dados disponibilizados na classe da respectiva entidade.

Após a chamada das funções, é necessário adicionar a entidade recém criada ao banco de dados. Como mencionado no item 4.1.6, as entidades do Modelador foram criadas na entrada `MODEL SPACE` da tabela de blocos do AutoCAD (`Block Table` / `Block Table Record`). O procedimento para efetuar esta adição de entidades ao `MODEL SPACE` é repetitivo e padronizado (Figura 4.3.1-1).

![Diagrama de funções](image)

**Figura 4.3.1-1 – Estrutura da função `append`.**
4.3.2 – Numeração Automática

A criação de uma estrutura de forma organizada implica numa numeração sequencial e lógica dos seus elementos. A numeração manual de uma estrutura de grande porte pode ser demorada e erros podem ocorrer devido a deslizes cometidos pelo usuário.

Para implementar uma numeração automática é necessária a criação de um objeto, sem representação gráfica, que possa armazenar estes dados. Como visto no item 4.1.6, os objetos que não possuem representação gráfica são armazenados no dicionário NOD (Named Object Dictionary) do AutoCAD. Dentro do dicionário NOD, é possível criar dicionários específicos para cada fim, organizando assim os objetos.

A numeração de cada uma das três entidades deve ser independente, sendo necessária assim a criação de uma variável para cada tipo de entidade. Este objeto deve ainda ser armazenado juntamente com o arquivo de desenho para que seja possível, em sessões de trabalho futuras, a continuidade da numeração automática.

Uma numeração automática inteligente deve conter dois tipos de variáveis. A primeira, um número inteiro, deve conter o valor do próximo número a ser utilizado para numerar a entidade. A segunda, deve conter um vetor dinâmico de números inteiros que servirá para armazenar os números que não estão sendo utilizados, ou seja, de entidades que foram apagadas. Estas variáveis podem então ser armazenadas em um objeto único chamado Config1. Este objeto deve ser criado de forma semelhante às entidades, diferenciando-se apenas pela derivação. Como não é necessária a implementação de representação gráfica, o objeto Config1 será um objeto puro, sendo derivado de AcDbObject.
4.3.2.1 – Definição do Objeto Config1

A composição do objeto **Config1** é relativamente simples. As variáveis e funções membros de **Config1** estão descritas na Figura 4.3.2.1-1.

### VARIÁVEIS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>curNodeN</td>
<td>Adesk::Int16</td>
<td>Número corrente da numeração automática da entidade <strong>Node</strong>.</td>
<td>Permanente</td>
</tr>
<tr>
<td>curBarN</td>
<td>Adesk::Int16</td>
<td>Número corrente da numeração automática da entidade <strong>Bar</strong>.</td>
<td>Permanente</td>
</tr>
<tr>
<td>curFaceN</td>
<td>Adesk::Int16</td>
<td>Número corrente da numeração automática da entidade <strong>Face</strong>.</td>
<td>Permanente</td>
</tr>
<tr>
<td>avNodeN</td>
<td>AcDbIntArray</td>
<td>Vetor contendo números não utilizados na numeração da entidade <strong>Node</strong>.</td>
<td>Permanente</td>
</tr>
<tr>
<td>avBarN</td>
<td>AcDbIntArray</td>
<td>Vetor contendo números não utilizados na numeração da entidade <strong>Bar</strong>.</td>
<td>Permanente</td>
</tr>
<tr>
<td>avFaceN</td>
<td>AcDbIntArray</td>
<td>Vetor contendo números não utilizados na numeração da entidade <strong>Face</strong>.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

### FUNÇÕES

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
</tbody>
</table>

Figura 4.3.2.1-1 – Lista de variáveis e funções membros de **Config1**.

As funções sobrecarregadas **dwgInFields** e **dwgOutFields** efetuam a gravação dos dados relativos à numeração no arquivo DWG juntamente com as entidades.
4.3.2.2 – Criação do Objeto Config1

A criação do objeto Config1 é efetuada pela função createConfig1 que acessa o dicionário NOD para criar e inicializar o objeto. Dentro do dicionário NOD, foi criado um dicionário chamado DATA_OBJJS e uma chave (key) chamada CONFIG1 para armazenar o objeto Config1 (Figura 4.3.2.2-1).

![Diagrama de Fluxo](image)

Figura 4.3.2.2-1 – Estrutura do NOD após a criação do objeto Config1.

A função createConfig1 é chamada quando o Modelador é carregado e quando um novo desenho é iniciado. A estrutura da função createConfig1 está descrita no fluxograma da Figura 4.3.2.2-2.

![Fluxograma](image)

Figura 4.3.2.2-2 – Estrutura da função createConfig1.
4.3.2.3 – Implementação da Numeração Automática nas Entidades

A numeração automática é utilizada para determinar o número de identificação da entidade que está sendo criada. Para cada entidade, foi criada uma função responsável por esta numeração. A estrutura da função é igual para as três entidades, com exceção da entidade Nó, onde não é necessário chamar a função de ajuste (textFit) para ajustar a posição do texto de identificação.

As funções implementadas para este fim foram setNodeid, setBarid e setFaceid, respectivamente para as entidades Node, Bar e Face. A estrutura básica destas funções está descrita a seguir, na Figura 4.3.2.3-1.

![Diagrama de fluxo das funções setNodeid, setBarid e setFaceid.](image)
4.3.3 – Reatores de Entidades

Reatores são objetos especiais destinados à monitoração de eventos ocorridos no ambiente do AutoCAD. Um evento, no contexto do AutoCAD, significa uma modificação aplicada a um objeto, o acionamento de um comando, uma modificação no banco de dados, etc. Para cada notificação, deve ser implementado um objeto reator (*Object Reactor*) acoplado à entidade ou outro objeto que emitirá a notificação. Vários tipos de notificações podem ser emitidas durante um evento; por exemplo, quando se move uma entidade pode-se receber uma notificação de movimentação, uma de invocação do comando *MOVE* do AutoCAD e outra de modificação no banco de dados.

Especificamente nas entidades, os reatores trabalham de forma a fornecer as notificações desejadas quando a entidade gera algum evento. Para complementar o gerenciamento da numeração automática das entidades, foi implementado um reator em cada uma das entidades com o objetivo de possibilitar ajustes no objeto *Config1* durante a invocação do comando *ERASE* do AutoCAD.

Como visto no item 4.3.2, o objeto *Config1* contém um vetor dinâmico que armazena os números não utilizados na numeração automática. Estes números são gerados quando elementos da estrutura são apagados e a sequência de criação é interrompida. Se estes números pertencentes às entidades apagadas não fossem levados em conta, aconteceriam interrupções na sequência numérica das entidades da estrutura.

Para prevenir esta interrupção, foi implementado o reator *erased*. A implementação deste reator é feita através da criação de uma classe derivada de *AcDbObject*. A classe criada implementa funções herdadas de *AcDbObject* que determinam os tipos de notificação que serão suportadas pela entidade. Durante a criação da entidade, este objeto é acoplado à entidade através da função do ObjectARX chamada *addPersistentReactor*.
que recebe como parâmetro apenas o ponteiro para a entidade a qual ele se acoplará. Como todo objeto sem representação gráfica, o reator também é armazenado no dicionário de objetos (NOD).

Figura 4.3.3-1 – Estrutura da função de notificação *erased*. 
Foram acopladas às entidades as classes ReactorNode, ReactorBar e ReactorFace, respectivamente. Nas três classes, foi implementada apenas a função de notificação erased que é acionada quando se exclui uma entidade (comando ERASE do AutoCAD) ou então, quando se desfaz uma exclusão através do comando UNDO do AutoCAD (Figura 4.3.3-1). A estrutura descrita na Figura 4.3.3-1 é semelhante para as três entidades.

Na entidade Bar, a função de notificação erased tem uma outra função importante para o funcionamento do Modelador. Quando as entidades do tipo Bar são apagadas, elas se encarregam de apagar ou não os Nós pertencentes às mesmas. Os Nós inicial e final da Barra somente serão apagados se a Barra em questão for a única da lista de Barras de cada Nó. Esta verificação e a exclusão dos Nós estão descritas no fluxograma da Figura 4.3.3-2.
4.3.4 – Reatores de Comandos

Para possibilitar operações de edição e cópia de entidades é necessária a monitoração dos eventos relativos aos comandos do AutoCAD. No caso específico deste reator, basta que se crie um objeto, derivado de AcEditorReactor, que implementará as funções de notificação desejadas. Para implementar este objeto e as funções de notificação, foi criada a classe CommandReactor. Esta classe define e implementa as funções e variáveis descritas na Figura 4.3.4-1.

<table>
<thead>
<tr>
<th>VARIÁVEL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nome</strong></td>
</tr>
<tr>
<td>cIBarIds</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNÇÕES</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Nome</strong></td>
</tr>
<tr>
<td>commandWillStart</td>
</tr>
<tr>
<td>commandEnded</td>
</tr>
<tr>
<td>commandCancelled</td>
</tr>
<tr>
<td>endDeepClone</td>
</tr>
</tbody>
</table>

Figura 4.3.4-1 – Lista de variáveis e funções membros de CommandReactor.

Em alguns casos, é necessário o “bloqueio” de algum tipo de entidade durante o acionamento de um comando específico. Por exemplo, quando se executa o comando COPY do AutoCAD sobre entidades do Modelador, somente Barras e Superfícies podem ser selecionadas. Neste caso, através de uma função de notificação de início de comando, pode-se “trancar” a camada (Layer) de desenho específica do Nó, para evitar sua seleção na tela de
desenho do AutoCAD. As funções de notificação, tanto para entidades quanto para comandos, não podem basear suas operações por uma ordem sequencial. Não é garantida a ordem em que as notificações são acionadas como, por exemplo, no caso onde uma entidade é apagada. Notificações ocorridas dentro de um mesmo contexto são acionadas numa sequência aleatória. A seguir, serão descritas as aplicações do reator de comandos nas operações de edição e cópia de entidades.

4.3.5 – Edição de Entidades

Para manter a flexibilidade da estrutura e a compatibilidade com a manipulação de entidades nativas do AutoCAD, o Modelador possibilita a aplicação de modificações sobre as entidades que representam os elementos estruturais. Estas modificações são aplicadas através de comandos do próprio AutoCAD.

A aplicação destes comandos efetua, de maneira geral, uma transformação espacial ou distorção na entidade. Esta transformação pode ser simples ou envolver alterações na geometria da entidade. A finalidade de permitir estas transformações é tornar possível a edição da estrutura pelo usuário. Assim, torna-se possível o reaproveitamento das estruturas e o ajuste visual de problemas ou erros detectados.

O relacionamento entre as entidades Node e Bar é essencial no que diz respeito à livre movimentação. Como cada Barra está conectada à outra através de um Nó, a movimentação conjunta torna-se um ponto bastante importante. No caso específico do Modelador, a entidade Bar é responsável pelo comportamento da entidade Node. A partir de uma edição sobre a entidade Bar, seus respectivos Nós devem comportar-se da maneira esperada pelo usuário. Todo este mecanismo deve ainda obedecer ao comportamento
estrutural de cada entidade assim como a coerência da conectividade da estrutura e a incidência de Barras.

Para efetuar os ajustes e alterações necessários após a execução de cada comando, foram utilizadas as funções de notificação e foi criada uma função de verificação de consistência de Nós e Barras de forma conjunta. A função `verifyConsist`, descrita na Figura 4.3.5-1, tem como objetivo ajustar a posição das Barras em função das transformações aplicadas aos Nós.

![Diagrama da função `verifyConsist`](image-url)

Figura 4.3.5-1 – Estrutura da função `verifyConsist`. 
A aplicação de uma função similar sobre as Barras poderia gerar um laço de repetição infinito devido a uma espécie de reação em cadeia. Se uma Barra é movida, por exemplo, a função para ajuste dos Nós seria chamada e moveria os Nós inicial e final para a nova posição da Barra. Neste caso, estes Nós ficariam fora da posição em relação às outras Barras conectadas a eles chegando-se então a um impasse de qual entidade teria a preferência para efetuar a correção, Nó ou Barra. Esta reação em cadeia ocorre porque uma Barra afeta outras Barras através de um Nó comum. A função verifyConsist é executada de forma correta porque faz com que os Nós tenham preferência para efetuar a correção. Isto é possível porque um Nó somente afeta um dos pontos de uma Barra, quando é alterada sua posição, gerando apenas um ajuste no comprimento da Barra pois o outro Nó da Barra permanece inalterado (Figura 4.3.5-2).

Situação normal.  
Ambas as Barras conectadas ao Nó 5.  
Barra 11 conectada também ao Nó 1.  
Barra 12 conectada também ao Nó 3.

Posição do Nó 5 alterada por algum comando de edição do AutoCAD.  
Notificação executada.  
Função de correção verifyConsist acionada.

Função verifyConsist executada.  
Pontos de extremidade das Barras 11 e 12 ajustados para a nova posição.  
Inclinação e posição do texto de identificação das Barras ajustadas.  
Posição dos Nós 1 e 3 inalterada.

Figura 4.3.5-2 – Exemplo de aplicação da função verifyConsist.
A aplicação de um comando de edição numa Barra isolada não causará nenhum efeito pois, quando a função de verificação `verifyConsist` for chamada, a Barra será movida para sua posição original. Em geral, os comandos de edição do AutoCAD causam uma movimentação das entidades. As funções de notificação de início, fim e cancelamento de comandos recebem como parâmetro o nome do comando que as chamou, ou seja, o comando que está sendo executado no AutoCAD.

Para os comandos de edição do AutoCAD, as funções de notificação executam as tarefas descritas na Figura 4.3.5-3. O congelamento de `Layers` (camadas) permite que seja desabilitada a seleção das entidades que utilizam a(s) camada(s) congelada(s) na tela gráfica do AutoCAD. Os comandos criados pelo Modelador, como `BAR`, `NREL`, etc., também podem ser incluídos nas funções de notificação sem qualquer problema. Os comandos acionados através do pontos de `GRIP` também devem ser controlados pois efetuam transformações nas entidades.

<table>
<thead>
<tr>
<th>Notificação</th>
<th>Comandos</th>
<th>Tarefas</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>commandWillStart</code></td>
<td>ERASE</td>
<td>Congelar <code>Layer</code> da entidade <code>Node</code>.</td>
</tr>
<tr>
<td></td>
<td>EXPLODE</td>
<td>Congelar <code>Layer</code> da entidade <code>Bar</code>.</td>
</tr>
<tr>
<td></td>
<td>MIRROR</td>
<td>Congelar <code>Layer</code> da entidade <code>Node</code>.</td>
</tr>
<tr>
<td></td>
<td>ERASE</td>
<td>Descongelar <code>Layer</code> da entidade <code>Node</code>.</td>
</tr>
<tr>
<td></td>
<td>EXPLODE</td>
<td>Descongelar <code>Layer</code> da entidade <code>Node</code>.</td>
</tr>
<tr>
<td></td>
<td>MOVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ROTATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>STRETCH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCALE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRIP_MOVE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRIP_ROTATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRIP_STRETCH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>GRIP_SCALE</td>
<td></td>
</tr>
<tr>
<td><code>commandEnded</code></td>
<td></td>
<td>Executar a função <code>verifyConsist</code>.</td>
</tr>
<tr>
<td></td>
<td>Todos os</td>
<td></td>
</tr>
<tr>
<td></td>
<td>comandos</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Descongelar <code>Layer</code> da entidade <code>Node</code>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Descongelar <code>Layer</code> da entidade <code>Bar</code>.</td>
</tr>
</tbody>
</table>

Figura 4.3.5-3 – Tarefas realizadas pelas funções de notificação durante a edição de entidades.
4.3.6 – Cópia de Entidades

Outra capacidade existente no AutoCAD, que aumenta de forma significativa o desempenho na criação de um desenho, é a de copiar entidades. No ambiente do AutoCAD, é possível efetuar cópias especificando pontos de precisão permitindo assim que as entidades copiadas possam ser dispostas da forma desejada.

O aumento de desempenho torna-se mais significativo ainda quando os desenhos apresentam algum tipo de simetria, mesmo que estejam em escalas diferentes. As entidades do Modelador também têm sua operação de cópia habilitada. Torna-se possível então, a criação de módulos de estrutura e o reaproveitamento destes módulos em outras partes do projeto.

A simples implementação da operação de cópia nas entidades do Modelador traria uma série de problemas que abririam espaço para a existência de erros cometidos pelo usuário. Além disso, com entidades numeradas, as entidades do Modelador precisam ter sua numeração ajustada de acordo com a operação de cópia. Mantendo a coerência com o comportamento conjunto das entidades Bar e Node, as operações de cópia somente foram implementadas nas entidades Bar e Face. A cópia da entidade Node é controlada e efetuada pela própria entidade Bar.

A operação de cópia das entidades Bar e Face foi realizada em dois estágios de implementação. O primeiro estágio determinou a numeração das entidades copiadas e o segundo estágio, o ajuste de posicionamento e a conectividade entre Nós e Barras após a operação de cópia, utilizando reatores de comandos.

A seguir, são descritos os dois estágios de forma detalhada.
4.3.6.1 – Numeração Após a Cópia

Quando as entidades são copiadas no AutoCAD, uma série de procedimentos são realizados. Como mencionado no item 4.2, as entidades possuem uma identificação única no banco de dados do AutoCAD (AcDbObjectId). Durante a operação de cópia, cada entidade selecionada é colocada numa espécie de mapeamento de identificações. Este mapeamento permite que sejam criadas novas entidades com as mesmas características das originais.

Para implementar uma característica especial à cópia das entidades do Modelador foi utilizada a função, derivada de AcDbEntity, deepClone. A função deepClone permite que o processo de mapeamento seja alterado para permitir que os dados copiados para a nova entidade sejam alterados ou ignorados. Quando o AutoCAD executa a função deepClone, já está disponível no mapeamento, a identificação que o banco de dados irá atribuir à nova entidade. De posse desta identificação, é possível acessar a nova entidade e alterar seus dados antes que a cópia seja concluída.

Redefinindo a função deepClone para as entidades Bar e Face pode-se chamar, na posição correta da função, a função membro definida para determinar a identificação da entidade (setBarId para a entidade Bar e setFaceld para a entidade Face), conforme descrito no item 4.2. Quando estas funções forem chamadas durante a operação de cópia, serão determinados os novos números de identificação das cópias da entidade original.

A Figura 4.3.6.1-1 descreve o procedimento realizado pela função deepClone durante a cópia da entidade. A estrutura da função é a mesma tanto para a entidade Bar quanto para a entidade Face, diferenciando-se apenas pela função que determina a numeração. No mapeamento (idMap) existem pares (idPair) de identificação onde a entidade original é chamada de key e a nova entidade a ser criada é chamada de value.
4.3.6.2 – Ajuste das Entidades Após a Cópia

Após a operação de cópia de entidades, alguns ajustes e verificações devem ser realizados para garantir a conectividade de Barras, a não sobreposição de Barras e Nós e a continuidade da numeração das entidades.
Estas verificações e ajustes são feitos através da análise das entidades copiadas. Inicialmente, um problema deve ser contornado. As entidades copiadas se tornam acessíveis somente após a finalização da operação de cópia. Para contornar este problema, será utilizada uma função de notificação para armazenar, em uma variável global, as identificações das entidades que estão sendo copiadas. Após o término da operação de cópia, através da utilização de um reator de comando, os ajustes e verificações serão efetuados nas entidades utilizando as identificações armazenadas na variável global durante a operação de cópia. As funções de notificação utilizadas na operação de cópia estão listadas na Figura 4.3.6.2-1.

<table>
<thead>
<tr>
<th>Notificação</th>
<th>Comandos</th>
<th>Tarefas</th>
</tr>
</thead>
<tbody>
<tr>
<td>commandWillStart</td>
<td>COPY</td>
<td>Congelar Layer da entidade Node.</td>
</tr>
<tr>
<td></td>
<td>ARRAY</td>
<td>Zerar o vetor que irá armazenar as</td>
</tr>
<tr>
<td></td>
<td>MIRROR</td>
<td>identificações das Barras copiadas.</td>
</tr>
<tr>
<td>commandEnded</td>
<td>COPY</td>
<td>Ajustar Barras, criar Nós, refazer</td>
</tr>
<tr>
<td></td>
<td>MIRROR</td>
<td>numeração de Barras.</td>
</tr>
<tr>
<td></td>
<td>ARRAY</td>
<td></td>
</tr>
<tr>
<td>commandCancelled</td>
<td>Todos os</td>
<td>Descongeloar Layer da entidade Node.</td>
</tr>
<tr>
<td></td>
<td>comandos</td>
<td>Descongeloar Layer da entidade Bar.</td>
</tr>
</tbody>
</table>

Figura 4.3.6.2-1 – Tarefas realizadas pelas funções de notificação durante a cópia de entidades.

Por motivos de simplificação, somente as entidades Bar e Node serão analisadas após a operação de cópia. Para cada Barra copiada serão verificados o posicionamento, a cópia de Barra sobre Barra, a conectividade com Nós existentes, o ajuste da numeração e o posicionamento do texto de identificação.

Primeiramente, a implementação da função de notificação endDeepClone é feita para armazenar as informações (identificações) das Barras que foram selecionadas pelo usuário para a operação de cópia. Como mencionado no item 4.3.6.1, as identificações das entidades que estão sendo copiadas ficam temporariamente contidas no mapeamento através de pares de
identificações. Iterando sobre este mapeamento, pode-se coletar as identificações das Barras, armazenando-as na variável global ciBarids (um vetor dinâmico de identificações). A função de notificação endDeepClone recebe como parâmetro o mapeamento (idMap) e sua estrutura está descrita na Figura 4.3.6.2-2.

A função de notificação endDeepClone é executada antes da função de notificação de fim de comandos (commandEnded). A montagem do vetor dinâmico contendo as identificações das Barras copiadas acontece na execução da função endDeepClone.

Após a notificação de fim da operação de cópia a função de notificação de fim de comando (para os comandos COPY, MIRROR e ARRAY) é chamada pelo AutoCAD. A função commandEnded executa dois trechos de verificação e ajustes das entidades Bar e Node.
Inicialmente, a camada (Layer) dos Nós é descongelada e configurada como corrente. No primeiro trecho, o vetor criado pela função `endDeepClone` é percorrido e cada identificação é usada para abrir as Barras copiadas e verificar, utilizando a função `verifyBar`, se estas foram copiadas sobre outras Barras. Além da verificação das Barras, este primeiro trecho cria, se necessário, novos Nós (Figura 4.3.6.2-3).

![Diagrama de fluxo](image)

**Figura 4.3.6.2-3 – Primeiro trecho da função de notificação `commandEnded`.**

Este primeiro trecho da função `commandEnded` possui uma estrutura similar à utilizada na criação de Barras e Nós (item 4.2.3), cobrindo as três possibilidades de criação.
Em alguns casos, as Barras serão copiadas sobre Barras existentes e, conforme visto na Figura 4.3.6.2-3, estas Barras duplicadas serão apagadas. Neste caso, ocorrerá uma descontinuidade na numeração sequencial das Barras. Para corrigir este problema, o segundo trecho da função `commandEnded` efetua a atualização da numeração após a exclusão das Barras duplicadas e ainda executa a função de verificação de consistência entre Barras e Nós (Figura 4.3.6.2-4).

![Diagrama de fluxo](image)

**Figura 4.3.6.2-4** – Segundo trecho da função de notificação `commandEnded`. 
4.3.7 – O Comando OFFSET

Uma das grandes facilidades encontradas, quando se utiliza a interface gráfica do AutoCAD, é a existência de comandos que tornam a criação de um projeto muito mais simples e dinâmica. Em especial, um destes comandos facilita de forma significativa a criação de desenhos baseados em malhas compostas por linhas, arcos ou curvas. O comando OFFSET permite que se crie uma entidade paralela a partir de uma distância da entidade original.

A facilidade no modelamento de uma estrutura composta por módulos fica bastante acentuada quando é possível a utilização de um comando como o OFFSET do AutoCAD. No caso do Modelador, a única entidade em que o OFFSET deve ser aplicado é a entidade Barra. No caso específico da aplicação do OFFSET sobre uma linha reta, o efeito deste comando é similar ao de uma cópia da linha numa direção perpendicular, criando-se assim outra linha de igual comprimento e inclinação no plano.

Como uma entidade “costumizada”, a entidade Bar deve sobrecarregar algumas funções que permitem a utilização do comando OFFSET sobre a Barra. Estas funções efetuam cálculos de derivadas, projeções e curvas paramétricas para calcular a posição da nova entidade criada pelo comando OFFSET. No caso de uma linha simples, como é o caso da Barra, estas funções são simplificadas sensivelmente. As funções a serem sobrecarregadas, listadas na Figura 4.2.1.2-1, são herdadas da classe AcDbCurve do ObjectARX e estão descritas a seguir:

- IsPlanar : Função que tem como objetivo verificar se a entidade é uma entidade plana. No caso da Barra, a função retorna Adesk::kTrue (verdadeiro) porque uma linha no espaço sempre está contida em um plano arbitrário. Além disso, a entidade Bar é uma entidade linear apesar de estar representada no espaço.
♦ getPlane: Função para determinar o plano da entidade. No caso da Barra, é necessário apenas retornar uma identificação de que a entidade é linear (AcDb::klinear).

♦ getStartParam: Função que retorna o parâmetro inicial da curva numa parametrização que vai de 0.0 até 1.0. No caso da Barra, o parâmetro inicial é igual a 0.0.

♦ getPointAtParam: Função que calcula, a partir de um parâmetro, o ponto espacial localizado na curva. Primeiro, é calculado o vetor da curva utilizando a subtração vetorial dos pontos inicial e final da Barra. A partir deste vetor, o ponto é calculado pela adição do ponto inicial da Barra com a multiplicação do vetor da curva pelo parâmetro.

♦ getFirstDeriv: Função que calcula a primeira derivada da curva que, no caso da Barra, é obtida pela subtração vetorial dos pontos inicial e final, ou seja, a própria curva.

♦ getClosestPointTo: Função que cria uma linha ou segmento de linha para determinar qual o ponto mais próximo sobre a curva, a partir de um ponto passado como parâmetro para a função.

♦ getOffsetCurves: Função que cria a nova entidade. Inicialmente, é calculado um vetor perpendicular à Barra. Os pontos inicial e final da Barra original são então transformados por uma matriz tridimensional em função do vetor perpendicular que representa a distância linear entre a entidade original e a nova entidade a ser criada. A partir destes novos pontos, é criada uma nova Barra e seus respectivos Nós utilizando as funções de verificação já definidas: verifyNode e verifyBar. Este procedimento é similar à criação de Barras descrita no item 4.2.3.
4.3.8 – Renumeração

A partir de alterações subsequentes em uma estrutura, podem acontecer desagrupamentos das numerações das entidades. Estes desagrupamentos se tornam bastante inconvenientes quando o projetista procura por algum elemento específico na estrutura. Caso este elemento procurado não esteja localizado numa região do desenho próxima a outras entidades com numeração semelhante, ou seja, números próximos, a procura deste elemento num projeto de grandes dimensões torna-se bastante difícil.

Um outro fator que pode gerar o desagrupamento das entidades com numeração próxima é a utilização de comandos de cópia. Como visto ao longo do item 4.3, as operações de cópia podem excluir entidades que se sobrepõem a outras existentes e são, por consequência, apagadas. No AutoCAD a cópia de entidades é feita de forma aleatória quanto à ordem de seleção. Quando o usuário utiliza uma janela para selecionar as entidades a serem copiadas, não existe ordem lógica para efetuar a operação de cópia nestas entidades e, muito menos, esta ordem é determinada pela numeração. Quando se copia grupos grandes de entidades, pode ocorrer um desagrupamento de numeração.

Nas empresas que trabalham com projetos, muitas vezes existem padrões de numeração de Nós, Barras e Superfícies. Cada empresa adota um critério que especifica que a numeração obedece uma direção especificada por um eixo orientado. Além disso, no caso das Barras, existem opções de numerar primeiramente as Barras na posição horizontal, depois as Barras na posição vertical e finalmente as Barras inclinadas.

Para implementar esta facilidade, o Modelador oferece o comando RENUM. Este comando permite que o usuário efetue uma renumeração nas entidades do Modelador especificando diversos parâmetros para cada entidade de forma independente. O comando é executado através do quadro de diálogo da Figura 4.3.8-1.
Para cada entidade, pode ser especificado o número inicial da renumeração. Este número permanece então como número mínimo para cada entidade criada. Este procedimento é realizado através do acesso ao objeto `Config1` de forma a atribuir o valor especificado no quadro de diálogo como o valor das variáveis `curNodeN`, `curBarN` e `curFaceN`. Além disso, os vetores de números não utilizados são zerados.

A renumeração é feita através de uma ordenação das identificações de cada entidade em um vetor a partir de um critério. Este critério é especificado de acordo com o tipo da entidade onde um ponto central é calculado para cada entidade. Na entidade `Node`, este ponto é o próprio ponto de inserção do Nó.
Na entidade **Face** o ponto de referência é o centro de gravidade da Superfície. Na entidade **Bar** a referência é o ponto médio da Barra. No caso das Barras esta ordenação ainda depende do critério de preferência do eixo escolhido no quadro de diálogo. A ordenação é feita através de rotinas sucessivas utilizando o método de ordenação tipo Bolha [Got93]. Este método analisa um conjunto de dados em um vetor e efetua uma comparação em pares trocando os elementos de posição de acordo com o critério. No final do processo, o conjunto está ordenado na forma especificada. A estrutura das funções **Renum_Nodes** e **Renum_Faces**, que aplica a ordenação em Nós e Superfícies respectivamente, está descrita na Figura 4.3.8.2. A estrutura é análoga diferenciando-se apenas pelo tipo de entidade.

![Diagrama de Funções](image)

**Figura 4.3.8-2 – Estrutura das funções Renum_Nodes e Renum_Faces.**
No caso específico das Barras, a rotina de ordenação é mais complexa devido à possibilidade de se efetuar a ordenação utilizando o critério de eixos preferenciais. A estrutura da função Renum_Bars está descrita na Figura 4.3.8-3.

Figura 4.3.8-3 – Estrutura da função Renum_Bars.
Ordenação pela coordenada Y dos pontos contidos em BarPts utilizando o critério de comparação especificado no quadro de diálogo pela direção do eixo Y (positivo ou negativo). Ordenar somente os pontos com a mesma coordenada Z. Ordenação do vetor BarIds em função do vetor BarPts.

Ordenação pela coordenada X dos pontos contidos em BarPts utilizando o critério de comparação especificado no quadro de diálogo pela direção do eixo X (positivo ou negativo). Ordenar somente os pontos com a mesma coordenada Y e Z. Ordenação do vetor BarIds em função do vetor BarPts.

Figura 4.3.8-4 – Rotina de ordenação sem considerar o eixo X ou Y.

Figura 4.3.8-5 – Rotina de ordenação considerando primeiro o eixo X.

Figura 4.3.8-6 – Rotina de ordenação considerando primeiro o eixo Y.

Em cada seqüência, são feitas verificações para preservar as ordenações efetuadas em um passo anterior da seqüência. No final dos três tipos de seqüências, as Barras estão ordenadas de acordo com os critérios especificados.
4.4 – Carregamentos

Definida a geometria da estrutura através das entidades **Node**, **Bar** e **Face**, podem ser definidos os carregamentos. Os tipos de carregamentos que podem incidir sobre uma estrutura são diversos. Cada elemento estrutural recebe um tipo de carga que possui características próprias. Além dos diversos tipos de carga, tem-se usualmente o agrupamento destas cargas em casos de carregamentos.

Os casos de carregamentos são utilizados para possibilitar a combinação de cargas gerando várias configurações de carregamento sobre a estrutura. A utilização de casos de carregamento permitirão ao calculista a simulação do comportamento da estrutura para estas diversas combinações.

O Modelador implementa inicialmente os seguintes carregamentos:

- Carga concentrada na entidade **Node** (direções **X**, **Y** e **Z**);
- Carga uniformemente distribuída na entidade **Face** (direção **Z**);
- Carga concentrada na entidade **Bar** (direção **Z**);
- Carga uniformemente distribuída parcial na entidade **Bar** (direção **Z**);
- Carga uniformemente distribuída completa na entidade **Bar** (direção **Z**).

No presente trabalho, foram implementadas a carga concentrada na entidade **Node**, a interface de implementação da carga distribuída na **Face** e os casos de carregamentos. A complementação da carga distribuída na entidade **Face** e as cargas na entidade **Bar** estão descritas em [Hüt98]. Além das cargas, os casos de carregamentos foram implementados para gerenciar todos os tipos de carregamento do Modelador através de um objeto.
4.4.1 – Casos de Carregamentos

O controle dos carregamentos através de um dispositivo que permite agrupar os vários tipos de carga é essencial para a organização de uma estrutura. Considerando a existência de apenas um grupo de casos de carregamentos em um arquivo de desenho do AutoCAD, pode-se implementar estes casos de carregamentos através de objetos no dicionário NOD do AutoCAD. Além de objetos que contenham as características de cada caso de carregamento (objeto **Casos**), deve existir também um outro objeto que armazena o caso em uso corrente no AutoCAD. Assim, cada carregamento criado utilizará o caso de carregamento corrente (objeto **CurrCaso**).

O gerenciamento destes casos de carregamentos possibilita o controle de cada carga em cada entidade. Através deste mecanismo, podem ser removidos casos inteiros de uma só vez. No item 4.5, são utilizadas ferramentas para possibilitar a exibição destas cargas e casos de carregamentos de forma independente.

Além da implementação destes objetos, foram criados, juntamente com as cargas, números de identificação que representam o caso de carregamento de cada carga existente. As informações relativas às cargas foram armazenadas sob a forma de variáveis membros nas entidades. Não foram criadas novas entidades para armazenar os dados relativos às cargas.

As informações relativas aos casos de carregamentos são armazenadas no próprio arquivo de desenho do AutoCAD possibilitando assim a permanência dos dados através das várias sessões de desenho.
4.4.2 – O Objeto Casos

O primeiro objeto implementado foi o objeto chamado Casos. Este objeto contém informações sobre cada caso de carregamento. Para cada caso de carregamento, serão armazenados um número de identificação, uma descrição do caso, uma variável do tipo boolean (falso ou verdadeiro) determinando se o caso está habilitado ou não e uma sigla que servirá como uma chave (key) no dicionário NOD.

Da mesma forma que o objeto de numeração, Config1, o objeto Casos não possui representação gráfica. Para cada caso de carregamento criado será criado um objeto Casos para armazenar as informações relativas a este caso de carregamento no dicionário do AutoCAD. O objeto Casos é criado a partir de uma derivação da classe AcDbObject, do ObjectARX, que permite a criação de novos objetos dentro do AutoCAD. Além das funções usualmente implementadas, foram criadas algumas funções utilitárias para facilitar o acesso às informações relativas a cada caso de carregamento e a manipulação destes casos no dicionário do AutoCAD.

As funções sobrecarregadas dwgInFields e dwgOutFields efetuam o armazenamento dos dados relativos aos objetos do tipo Casos no arquivo de desenho do AutoCAD (DWG). A numeração de cada caso de carregamento é automaticamente gerada pela rotina que cria os objetos do tipo Casos. Esta numeração tem como objetivo minimizar os dados armazenados nas entidades diminuindo assim o tamanho do arquivo DWG e a utilização de memória.

A seguir, são descritas as etapas de definição, criação e manipulação do objeto Casos, através da classe Casos e das respectivas variáveis, funções sobrecarregadas e funções utilitárias.
4.4.2.1 – Definição do Objeto Casos

As variáveis, funções membros e funções utilitárias da classe **Casos** estão descritas a seguir na Figura 4.4.2.1-1.

### VARIÁVEIS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>Num</td>
<td>Adesk::Int16</td>
<td>Número de identificação de cada caso de carregamento.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Desc</td>
<td>char *</td>
<td>Descrição do caso de carregamento.</td>
<td>Permanente</td>
</tr>
<tr>
<td>Enabled</td>
<td>Adesk::Boolean</td>
<td>Informação sobre a habilitação do caso de carregamento.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

### FUNÇÕES MEMBROS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>setNum</td>
<td>void</td>
<td>Armazenar o número de identificação do caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>getNum</td>
<td>Adesk::Int16</td>
<td>Retornar o número de identificação do caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>setDesc</td>
<td>void</td>
<td>Armazenar a descrição do caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>getDesc</td>
<td>char *</td>
<td>Retornar a descrição do caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>setEnabled</td>
<td>void</td>
<td>Habilitar ou desabilitar o caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>getEnabled</td>
<td>Adesk::Boolean</td>
<td>Retornar o estado de habilitação do caso.</td>
<td>Específica</td>
</tr>
</tbody>
</table>
### FUNÇÕES UTILITÁRIAS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>createCasos</td>
<td>int</td>
<td>Cria um objeto Casos com os parâmetros especificados.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCasso</td>
<td>void</td>
<td>Retornar as informações de um caso a partir de sua sigla.</td>
<td>Específica</td>
</tr>
<tr>
<td>removeEntityCasos</td>
<td>void</td>
<td>Remover em todas as entidades as cargas relacionadas com o caso especificado.</td>
<td>Específica</td>
</tr>
<tr>
<td>delCasso</td>
<td>void</td>
<td>Excluir um caso a partir de sua sigla.</td>
<td>Específica</td>
</tr>
<tr>
<td>modifyCasso</td>
<td>void</td>
<td>Modificar as informações de um caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>searchAvNumber</td>
<td>Adesk::Boolean</td>
<td>Verificar se um número já está sendo utilizado por algum caso.</td>
<td>Específica</td>
</tr>
<tr>
<td>getAvNumber</td>
<td>int</td>
<td>Retornar o primeiro número disponível para a identificação do novo caso entre 1 e 100.</td>
<td>Específica</td>
</tr>
<tr>
<td>isEnabled</td>
<td>Adesk::Boolean</td>
<td>Verificar se um caso está habilitado ou não.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCassoName</td>
<td>char *</td>
<td>Retornar o nome de um caso a partir de seu número.</td>
<td>Específica</td>
</tr>
<tr>
<td>getNumberOfCasos</td>
<td>int</td>
<td>Retornar o número de objetos do tipo Casos existentes.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

Figura 4.4.2.1-1 – Lista de variáveis, funções membros e funções utilitárias do objeto Casos.

### 4.4.2.2 – Criação do Objeto Casos

A criação do objeto Casos é efetuada pela função createCasos que acessa o dicionário NOD para criar um objeto Casos com o número de
identificação um, a descrição "DEFAULT", habilitado e com a sigla "DEFAULT". Dentro do dicionário NOD, foi criado um dicionário chamado CASOS_OBJJS onde cada chave (key) é a sigla do objeto Casos (Figura 4.4.2.2-1).

![Diagrama de NOD e CASOS_OBJJS](image)

Figura 4.4.2.2-1 – Estrutura do NOD após a criação de 4 objetos Casos.

A função `createCasos` é chamada quando o Modelador é carregado e quando um novo desenho é iniciado. A estrutura da função `createCasos` está descrita no fluxograma da Figura 4.4.2.2-2.

```
<table>
<thead>
<tr>
<th>NOD contém CASOS_OBJJS?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Criar uma chave a partir da sigla.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Atribuir o número de identificação à variável Num. Atribuir a descrição à variável Desc. Atribuir a habilitação à variável Enabled.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CASOS_OBJJS contém esta sigla?</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIM</td>
</tr>
<tr>
<td>Criar um objeto Casos.</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Criar o dicionário CASOS_OBJJS</td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
```

Figura 4.4.2.2-2 – Estrutura da função `createCasos`. 
4.4.2.3 – Manipulação do Objeto Casos

A manipulação dos objetos do tipo **Casos** é efetuada através de funções utilitárias. Cada função efetua uma tarefa utilizando o dicionário do AutoCAD (NOD) para acessar os objetos. A seguir, são apresentadas as estruturas de cada função utilitária.

**a)** *getCaso*: A partir da sigla do objeto **Casos**, esta função acessa o dicionário do AutoCAD e recupera as informações do objeto (Figura 4.4.2.3.a).

![Diagrama de fluxo para getCase](image)

Figura 4.4.2.3.a – Estrutura da função **getCaso**.

**b)** *removeEntityCases*: A partir do número de identificação do objeto **Casos**, esta função acessa a tabela *Block Table* do AutoCAD e chama, para cada tipo de entidade, a função que remove todas as cargas que estão relacionadas com este caso de carregamento (Figura 4.4.2.3.b). No item 4.4.4, as funções queremovem as cargas nas entidades serão detalhadas.
c) delCaso: A exclusão de um caso de carregamento implica na exclusão do objeto Casos correspondente e da remoção das cargas existentes nas entidades que estão relacionadas a este caso de carregamento. Antes de se apagar o objeto Casos, armazena-se o número de identificação deste caso de carregamento em uma variável que será passada para a função removeEntityCasos como parâmetro. A estrutura da função delCaso está descrita a seguir na Figura 4.4.2.3.c.
d) **modifyCaso**: Para modificar os dados do objeto **Casos**, basta que se passe os parâmetros para a função **modifyCaso**. A partir da sigla, o objeto é aberto e alterado (Figura 4.4.2.3.d).
e) searchAvNumber: Esta função verifica se o número passado como parâmetro ainda não foi utilizado por nenhum caso de carregamento (Figura 4.4.2.3.e).

f) getAvNumber: Função que verifica o menor número disponível, entre 1 e 100, para ser utilizado como identificação do novo objeto Casos a ser criado. Ao encontrar o número disponível, a função é interrompida e o valor deste número é retornado. A estrutura da função está descrita na Figura 4.4.2.3.f.
g) isEnabled: Função que verifica se o caso de carregamento, cujo número de identificação é passado como parâmetro, está ou não habilitado (Figura 4.4.2.3.g).

h) getCaseName: Função que, a partir do número de um caso de carregamento, retorna sua sigla (Figura 4.4.2.3.h).
i) `getNumberOfCAsos`: Função que abre a entrada `CASOS_OBJJS` do dicionário e retorna o número de objetos contidos (Figura 4.4.2.3.i).

![Diagrama de estrutura](image)

*Figura 4.4.2.3.i – Estrutura da função `getNumberOfCAsos`.*

### 4.4.3 – O Objeto CurrCAso

O segundo objeto criado para controlar os casos de carregamentos tem como objetivo armazenar os dados relativos ao caso de carregamento que está em uso. A funcionalidade proporcionada pelo objeto `CurrCAso` permite que as cargas criadas sejam atribuídas ao caso de carregamento corrente.

Neste objeto são armazenados o número e a sigla do caso de carregamento corrente. O objeto `CurrCAso` é criado a partir de uma derivação da classe `AcDbObject` herdando somente as funções de manipulação do arquivo DWG do AutoCAD. A seguir, são descritas as etapas de definição, criação e manipulação do objeto `CurrCAso`.

#### 4.4.3.1 – Definição do Objeto CurrCAso

As variáveis, funções membros e funções utilitárias da classe `CurrCAso` estão descritas a seguir na Figura 4.4.3.1-1.
### VARIÁVEIS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>Curr</td>
<td>Adesk::Int16</td>
<td>Número de identificação do caso de carregamento corrente.</td>
<td>Permanente</td>
</tr>
<tr>
<td>CurrName</td>
<td>char *</td>
<td>Nome do caso de carregamento corrente.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

### FUNÇÕES MEMBROS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>dwgInFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a leitura dos dados a partir do arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>dwgOutFields</td>
<td>Acad::ErrorStatus</td>
<td>Efetuar a escrita dos dados no arquivo DWG.</td>
<td>Sobrecarregada</td>
</tr>
<tr>
<td>setCurr</td>
<td>void</td>
<td>Armazenar o número de identificação do caso corrente.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCurr</td>
<td>Adesk::Int16</td>
<td>Retornar o número de identificação do caso corrente.</td>
<td>Específica</td>
</tr>
<tr>
<td>setCurrName</td>
<td>void</td>
<td>Armazenar a sigla do caso corrente.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCurrName</td>
<td>char *</td>
<td>Retornar a sigla do caso corrente.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

### FUNÇÕES UTILITÁRIAS

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>createCurrCaso</td>
<td>void</td>
<td>Cria um objeto CurrCaso.</td>
<td>Específica</td>
</tr>
<tr>
<td>setCurrCaso</td>
<td>void</td>
<td>Tornar um caso corrente.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCurrCaso</td>
<td>char *</td>
<td>Retornar a sigla do caso corrente.</td>
<td>Específica</td>
</tr>
<tr>
<td>getCurrCasoNum</td>
<td>int</td>
<td>Retornar o número do caso corrente.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

Figura 4.4.3.1-1 – Lista de variáveis, funções membros e funções utilitárias do objeto CurrCaso.
4.4.3.2 – Criação do Objeto CurrCaso

A criação do objeto CurrCaso é efetuada pela função createCurrCaso que acessa o dicionário NOD para criar um objeto CurrCaso com o número de identificação um e o nome “DEFAULT” de acordo com o caso criado no item 4.4.2. Dentro do dicionário NOD, foi criado um dicionário chamado CASOS_SET e uma chave (key) CURRENT_CASO (Figura 4.4.3.2-1).

![Diagrama de NOD](image)

Figura 4.4.3.2-1 – Estrutura do NOD após a criação do objeto CurrCaso.

A função createCurrCaso é chamada quando o Modelador é carregado e quando um novo desenho é iniciado. A estrutura da função createCurrCaso está descrita no fluxograma da Figura 4.4.3.2-2.

![Fluxograma de createCurrCaso](image)

Figura 4.4.3.2-2 – Estrutura da função createCurrCaso.
4.4.3.3 – Manipulação do Objeto CurrCaso

A manipulação do objeto **CurrCaso** é efetuada através de funções utilitárias. Cada função efetua uma tarefa utilizando o dicionário do AutoCAD (NOD) para acessar este objeto. A seguir, são apresentadas as estruturas de cada função utilitária.

**a) setCurrCaso**: A partir da sigla do objeto **Casos** e do seu número de identificação, esta função acessa o dicionário do AutoCAD e faz com que o objeto **CurrCaso** contenha as informações passadas como parâmetro (Figura 4.4.3.3.a).

![Diagrama de fluxo do setCurrCaso](image)

Figura 4.4.3.3.a – Estrutura da função **setCurrCaso**.

**b) getCurrCaso**: Função que acessa o objeto **CurrCaso** contido na entrada **CURRENT_CASO**, dentro do dicionário NOD, retornando a sigla do caso de carregamento corrente (Figura 4.4.3.3.b).
c) getCurrCaseNum: Função que acessa o objeto **CurrCase** contido na entrada **CURRENT_CASO**; dentro do NOD, retornando o número de identificação do caso de carregamento corrente (Figura 4.4.3.3.c).
4.4.4 – A função removeCaso

Quando um caso de carregamento é removido, torna-se necessária a remoção das cargas que estão ligadas a este caso de carregamento. Como visto na função delCaso, item 4.4.2.3, existe uma função chamada removeCaso presente em cada uma das três entidades do Modelador.

Nas entidades, as cargas são organizadas em vetores dinâmicos. Inicialmente, existe um vetor que armazena o valor da carga. Paralelamente a este primeiro vetor, existem outros, de acordo com o tipo de carga, contendo informações adicionais tais como distância do apoio, eixo de aplicação, tamanho, etc. Finalmente, existe um último vetor paralelo que armazena a codificação relativa ao caso de carregamento que a carga em questão se relaciona.

A remoção de todas as cargas aplicadas a uma das três entidades ocorre em função da análise do vetor de identificações de casos de carregamento em cada tipo de carga. Por exemplo, quando as cargas de uma Barra forem removidas, esta remoção será feita através de uma análise de cada vetor relativo a cada tipo de carga. Serão removidas aquelas cargas que contiverem, no vetor paralelo de casos de carregamentos, um número de identificação igual àquele passado como parâmetro para a função removeCaso.

Um ponto importante a ser ressaltado é que, quando se trabalha com vetores dinâmicos, existe um detalhe que ocorre quando uma posição é removida. Quando este vetor está sendo percorrido, numa iteração unitária, ou seja, de uma em uma posição, ao se remover uma posição o comprimento deste vetor é automaticamente ajustado. Com este encurtamento do vetor é necessário que se volte uma posição na variável de incremento de posições. Se isto não for feito, a posição que estava colocada após a posição removida vai ocupar esta posição e não será considerada na próxima verificação. O
mecanismo básico da função membro removeCaso está descrito a seguir, na Figura 4.4.4-1.

![Diagrama de fluxo](image)

**Figura 4.4.4-1 – Estrutura básica da função removeCaso.**

### 4.4.5 – O Comando CASOS

Para efetuar o gerenciamento, criação, remoção e edição dos casos de carregamento, o Modelador disponibiliza o comando CASOS. A partir do acionamento deste comando é exibido um quadro de diálogo (Figura 4.4.5-1) contendo todas as informações relativas aos casos de carregamento. Utilizando um processo de transações do AutoCAD, é possível efetuar um cancelamento de todas as operações realizadas neste quadro de diálogo, inclusive remoção de casos de carregamento, quando o usuário aciona o botão
CANCEL para sair do quadro. Neste caso, as operações são canceladas e o desenho do AutoCAD retorna ao mesmo estado encontrado antes do acionamento do comando CASOS.

Figura 4.4.5-1 – Quadro de diálogo do comando CASOS.

Para se criar um novo caso de carregamento basta que o usuário acione o botão NEW e um quadro de diálogo será mostrado para a entrada dos dados deste novo caso de carregamento (Figura 4.4.5-2). Neste quadro é possível especificar a descrição do novo caso de carregamento, sua sigla e se este caso está ou não habilitado. Cada caso de carregamento deve ter uma sigla única pois esta é usada como a chave (key) no dicionário de objetos do AutoCAD (NOD) onde cada entrada deve ter um nome único. O Modelador determina automaticamente um número de identificação para cada novo caso.
de carregamento criado. Este número é usado para referenciar cada caso de carregamento nos vetores de cargas das entidades.

Figura 4.4.5-2 – Quadro de diálogo para a criação de novos casos de carregamento.

Para se efetuar modificações quanto à descrição e habilitação de cada caso de carregamento, basta que se acione o botão MODIFY no quadro da Figura 4.4.5-1. Antes, deve ser selecionada a sigla, na lista de casos, daquele caso que se deseja modificar (Figura 4.4.5-3). O quadro de modificação é semelhante ao quadro de criação de um novo caso. A modificação de um caso de carregamento limita-se a modificação de sua descrição e habilitação devido à complexidade de se alterar o número de identificação deste caso. Esta numeração tem função somente no programa, não é necessário que o usuário se preocupe com este detalhe. A sigla também não pode ser alterada devido à sua função de chave (key) do dicionário do AutoCAD (NOD).
Figura 4.4.5-3 – Quadro de diálogo para a modificação de casos de carregamento.

Através do quadro de diálogo principal também é possível escolher qual caso de carregamento será configurado como corrente (Figura 4.4.5-4).

Figura 4.4.5-4 – Quadro de diálogo para escolha do caso de carregamento corrente.
Ao utilizar os quadros de diálogo do comando **CASOS**, o usuário receberá alguns avisos e mensagens de erro para prevenir operações ilegais ou que afetarão as entidades. Estas mensagens são mostradas a seguir com suas respectivas descrições (Figura 4.4.5-5).

**Figura 4.4.5-5 – Mensagens dos quadros de diálogo do comando **CASOS**.**
4.4.6 – Carga Concentrada na Entidade Node

A entidade **Node** pode receber dois tipos básicos de carregamento, a carga concentrada e o momento. No Modelador, somente foi implementada a carga concentrada, nas três direções principais X,Y e Z. A implementação do momento fletor pode ser efetuada numa etapa futura sem muitas dificuldades.

A carga concentrada na entidade **Node** pode ser aplicada no sentido positivo ou negativo de cada um dos três eixos. Além disso, cada carga pode ser atribuída a um caso de carregamento específico. A implementação da carga concentrada foi feita através da inclusão de novas variáveis e funções para manipulação que permitem a associação das cargas a cada entidade.

Foram implementados três vetores para armazenar o valor da carga, um código de identificação do eixo de aplicação e um código de identificação do caso de carregamento associado a cada carga. As funções implementadas permitem a manipulação das cargas aplicadas para inclusão e exclusão individual das mesmas. Estes vetores são dinâmicos e permitem que várias cargas sejam aplicadas em um mesmo eixo, numa mesma direção e em um mesmo caso de carregamento. Assim, o usuário pode controlar individualmente a inclusão de cada carga assim como os valores totais em cada eixo.

A seguir, são detalhadas as variáveis e funções acrescentadas à entidade **Node** para efetuar a implementação da carga concentrada.
4.4.6.1 – Variáveis e Funções

As variáveis e funções membros implementadas na classe `Node` estão descritas a seguir na Figura 4.4.6.1-1.

<table>
<thead>
<tr>
<th>VARIÁVEIS</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenterLoad</td>
<td>AcGeDoubleArray</td>
<td>Vetor que armazena os valores de cada carga concentrada.</td>
<td>Permanente</td>
</tr>
<tr>
<td>CenterLoadDir</td>
<td>AcDbIntArray</td>
<td>Vetor que armazena o código de identificação da direção de cada carga concentrada.</td>
<td>Permanente</td>
</tr>
<tr>
<td>CenterLoadCas</td>
<td>AcDbIntArray</td>
<td>Vetor que armazena o código de identificação do caso de carregamento associado a cada carga concentrada.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNÇÕES MEMBROS</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>insertCLoad</code></td>
<td>void</td>
<td>Inserir uma carga no vetor de cargas concentradas.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>removeCLoad</code></td>
<td>int</td>
<td>Remover uma carga do vetor de cargas concentradas.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>removeCaso</code></td>
<td>void</td>
<td>Remover todas as cargas associadas a um determinado caso.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>getLoadCas</code></td>
<td>int</td>
<td>Retornar código do caso de carregamento de uma carga.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>getLoadDir</code></td>
<td>int</td>
<td>Retornar o código do eixo de aplicação da carga.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>getLoadVal</code></td>
<td>double</td>
<td>Retornar o valor da carga.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>getLoadValLen</code></td>
<td>int</td>
<td>Retornar o comprimento do vetor de cargas concentradas.</td>
<td>Específica</td>
</tr>
<tr>
<td><code>drawLoad</code></td>
<td>void</td>
<td>Desenhar uma carga em um eixo e numa direção especificados.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

Figura 4.4.6.1-1 – Lista de variáveis e funções membros adicionados à entidade `Node` para implementar a carga concentrada.
A manipulação dos dados é feita através das funções membros que acessam as variáveis de acordo com os objetivos de cada função. A seguir, são descritas as estruturas de cada função membro.

a) insertCLoad: São passados como parâmetros para esta função o valor da carga concentrada, o código de identificação do eixo de aplicação e o caso de carregamento a que esta carga se associa. Esta função simplesmente adiciona cada parâmetro ao final de cada vetor correspondente.

b) removeCLoad: Esta função, a partir do valor, direção de aplicação e caso de carregamento, remove uma carga específica do vetor de cargas concentradas. Os três parâmetros são necessários para diferenciar cada carga. A estrutura da função está descrita na figura 4.4.6.1-2.

![Diagrama de função removeCLoad](image.png)

*Figura 4.4.6.1-2 – Estrutura da função removeCLoad.*
c) removeCasos: Conforme descrita no item 4.4.4, esta função tem como objetivo procurar e remover todas as cargas associadas a um caso de carregamento especificado. No caso da carga concentrada, esta busca é realizada somente no três vetores que armazenam as cargas concentradas e seus dados adicionais (direção e caso).

d) getLoadCas: A partir de uma posição do vetor de casos de carregamento, esta função retorna o código de identificação do caso de carregamento associado à carga concentrada contida no vetor paralelo na mesma posição.

e) getLoadDir: A partir de uma posição do vetor de direção de cargas, esta função retorna o código de identificação da direção (eixo) em que a carga em questão está aplicada. Os códigos são os seguintes:

- 1: eixo X positivo;
- 2: eixo X negativo;
- 3: eixo Y positivo;
- 4: eixo Y negativo;
- 5: eixo Z positivo;
- 6: eixo Z negativo.

f) getLoadVal: A partir de uma posição do vetor de cargas, esta função retorna o valor (número real) da carga concentrada aplicada na entidade Node. Este valor é um número real do tipo double armazenado através do vetor dinâmico do tipo AcGeDoubleArray.

g) getLoadValLen: Esta função tem como objetivo retornar o comprimento do vetor de cargas concentradas. Como os três vetores são paralelos, ou seja, são manipulados ao mesmo tempo, este valor é válido para qualquer um dos três vetores. Este dado é necessário para as rotinas que percorrem estes vetores.
h) drawLoad: Esta função tem como objetivo desenhar no AutoCAD uma representação das cargas concentradas em cada eixo e direção especificados. A partir do código de identificação da direção, esta função desenha, utilizando as primitivas gráficas do AutoCAD, um tipo de "seta" na direção e eixo especificados. Mesmo que existam mais de uma carga em um eixo e direção determinados somente uma "seta" é desenhada. O objetivo desta representação gráfica é indicar que existe pelo menos uma carga concentrada aplicada na direção e eixo especificados (Figura 4.4.6.1-3).

Figura 4.4.6.1-3 – Representação gráfica da carga concentrada no Nó.
4.4.6.2 – O Comando CLNODE

Para efetuar a aplicação de cargas concentradas nas entidades do tipo Node, o Modelador disponibiliza o comando CLNODE. Este comando permite a seleção individual ou conjunta de entidades do tipo Node. Quando a seleção contém mais de uma entidade Node, o comando CLNODE é executado através do prompt do AutoCAD. Neste caso, somente é permitida a inclusão, de uma só vez, de carregamentos. A seguir, é descrita a sequência de mensagens e perguntas exibidas no prompt do AutoCAD (Figura 4.4.6.2-1).

![Prompt do comando CLNODE quando mais de uma entidade Node é selecionada.](image)

Quando apenas uma entidade é selecionada, o comando CLNODE ativa um quadro de dia Aberto que permite ao usuário criar, modificar e remover cargas concentradas na entidade Node. Este quadro exibe as informações completas das cargas concentradas presentes na entidade, incluindo os casos de carregamentos presentes. É possível, através deste quadro, acessar cada elemento dos vetores dinâmicos contendo o valor das cargas. Estes vetores são filtrados em três listas, uma para cada eixo, incluindo um campo que informa o total das cargas em uma direção específica. As informações de outro caso de carregamento podem ser facilmente acessadas, dentro do próprio quadro, efetuando-se apenas uma escolha numa caixa de opções. As informações são atualizadas na entidade assim que o botão OK é acionado. Da
mesma forma que o comando **CASOS**, o comando **CLNODE** permite um cancelamento antes que as modificações sejam aplicadas à entidade. Este procedimento é implementado utilizando o mecanismo de transações do AutoCAD (Figura 4.4.6.2-2).

![Figura 4.4.6.2-2 - Quadro de diálogo do comando CLNODE quando somente uma entidade Node é selecionada.](image)

Antes de se acionar um dos três botões (**INSERT**, **REMOVE** ou **MODIFY**), o usuário deve clicar na lista de cargas na direção desejada. Ao selecionar a lista, o total de cargas na direção escolhida é mostrado na parte inferior. Para inserir uma nova carga, basta selecionar a lista e acionar o botão **INSERT**. Para remover um carga, basta selecionar a carga desejada e acionar o botão **REMOVE**. Ao selecionar a opção de remoção da carga, o usuário será avisado através de um pedido de confirmação do comando. Para modificar uma carga, basta selecionar uma carga de uma das três listas, acionar o botão
MODIFY e editar seu valor. A qualquer momento o caso de carregamento pode ser modificado. Cada carga criada, removida ou modificada afetará apenas o caso de carregamento em destaque no quadro de diálogo (Figura 4.4.6.2-3).

Figura 4.4.6.2-3 – Quadros de diálogo e mensagem do comando CLNODE.

4.4.6.3 – Cópia de Cargas Concentradas

Quando uma entidade Node é copiada, seus carregamentos não são copiados de forma automática. Este fato ocorre porque a entidade Node não implementa a função deepClone de cópia da entidade. Para que os dados relativos às cargas concentradas sejam copiados durante a operação de cópia, é necessária a inclusão manual da cópia durante o reator de comandos CommandEnded.
Os dados relativos aos carregamentos são recuperados a partir dos Nóspertencentes às Barras que estão sendo copiadas. Como visto no item 4.3.6,os Nóssão copiados pelas Barras. Assim, quando a notificação de fim de comando é acionada os dados podem ser resgatados das Barras originais ecopiados para os novos Nóss, que estão sendo criados.

A justificativa para a não implementação da função `deepClone` para aentidade `Node` é a abertura para futuras implementações de controle da cópiade carregamentos. Assim, pode-se implementar uma opção para o usuário decopiar ou não os carregamentos da entidade `Node` durante a operação decópia.

**4.4.7 – Carga Distribuída na Entidade Face**

Na entidade `Face`, foi implementado somente a carga distribuída superficial em uma única direção, o eixo Z. Por padrão, esta carga tem o valor negativo acompanhando a orientação da carga concentrada no Nó.

A implementação desta carga foi feita de forma semelhante à carga concentrada na entidade `Node`, com a simplificação de apenas um eixo deaplicação. Neste caso, a utilização de um terceiro vetor dinâmico paraarmazenar as informações sobre a direção de aplicação não foi necessária. Asfunções implementadas para manipular e controlar a carga distribuída naSuperfície são análogas às utilizadas na entidade `Node`.

A seguir, são listadas as variáveis e funções utilizadas naimplementação da carga distribuída na entidade `Face`. 
4.4.7.1 – Variáveis e Funções

As variáveis e funções membros implementadas na classe **Face** estão descritas a seguir na Figura 4.4.7.1-1.

<table>
<thead>
<tr>
<th>VARIÁVEIS</th>
<th>Nome</th>
<th>Tipo</th>
<th>Descrição</th>
<th>Duração</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ComLoad</td>
<td>AcGeDoubleArray</td>
<td>Vetor que armazena os valores de cada carga distribuída.</td>
<td>Permanente</td>
</tr>
<tr>
<td></td>
<td>ComLoadCas</td>
<td>AcDbIntArray</td>
<td>Vetor que armazena o código de identificação do caso de carregamento associado a cada carga distribuída.</td>
<td>Permanente</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FUNÇÕES MEMBROS</th>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>setLoad</td>
<td>void</td>
<td>Inserir uma carga no vetor de cargas distribuídas.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>getLoad</td>
<td>int</td>
<td>Retornar o valor de uma carga em uma posição especificada no vetor de cargas distribuídas.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>removeLoadCom</td>
<td>void</td>
<td>Remover uma carga distribuída específica.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>removeCaso</td>
<td>void</td>
<td>Remover todas as cargas associadas a um determinado caso.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>getLoadCas</td>
<td>int</td>
<td>Retornar código do caso de carregamento de uma carga.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>getLoadValLen</td>
<td>int</td>
<td>Retornar o comprimento do vetor de cargas distribuídas.</td>
<td>Especifica</td>
</tr>
<tr>
<td></td>
<td>drawLoad</td>
<td>void</td>
<td>Desenhar a carga distribuída.</td>
<td>Especifica</td>
</tr>
</tbody>
</table>

Figura 4.4.7.1-1 – Lista de variáveis e funções membros adicionados à entidade **Face** para implementar a carga distribuída.

Maiores detalhes sobre a implementação da entidade **Face** e de suas variáveis e funções podem ser encontrados em [Hüt98].
4.4.7.2 – O Comando CLFACE

A aplicação de cargas distribuídas em Superfícies é efetuada acionando-se o comando CLFACE do Modelador. Este comando permite a seleção individual ou conjunta de entidades do tipo Face. Quando a seleção contém mais de uma Superfície, o comando CLFACE é executado através do prompt do AutoCAD. Neste caso, somente é permitida a inclusão, de uma só vez, de carregamentos. Na Figura 4.4.7.2-1, é descrita a sequência de mensagens e perguntas exibidas no prompt do AutoCAD.

![Image of CLFACE command prompt with selected entities and load](image)

Figura 4.4.7.2-1 – Prompt do comando CLFACE quando mais de uma entidade Face é selecionada.

Quando apenas uma entidade é selecionada, o comando CLFACE ativa um quadro de diálogo que permite ao usuário criar, modificar e remover cargas. O quadro exibe as informações completas das cargas distribuídas presentes na entidade, incluindo os casos de carregamentos presentes. As informações de outro caso de carregamento podem ser facilmente acessadas dentro do próprio quadro efetuando-se apenas uma escolha numa caixa de opções. As informações são atualizadas na entidade assim que o botão OK é acionado. Da mesma forma que o comando CASOS e CLNODE, o comando CLFACE permite um cancelamento antes que as modificações sejam aplicadas à entidade. Este procedimento é implementado utilizando o mecanismo de transações do AutoCAD (Figura 4.4.7.2-2).
Antes de se acionar um dos três botões (INSERT, REMOVE ou MODIFY), o usuário deve clicar na lista de cargas. No canto superior direito do quadro, o total das cargas distribuídas é exibido. Para inserir uma nova carga basta selecionar a lista e acionar o botão INSERT. Para remover um carga, basta selecionar a carga desejada e acionar o botão REMOVE. Ao selecionar a opção de remoção da carga, o usuário será avisado através de um pedido de confirmação do comando. Para modificar uma carga, basta selecionar uma carga na lista, acionar o botão MODIFY e editar seu valor. A qualquer momento, o caso de carregamento pode ser modificado. Cada carga criada, removida ou modificada afetará apenas o caso de carregamento em destaque no quadro de diálogo (Figura 4.4.7.2-3).
**4.4.7.3 – Cópia de Cargas Distribuídas**

Quando uma entidade **Face** é copiada, seus dados de carregamento sãoautomaticamente copiados através da função **deepClone**. A implementação desta função habilita o AutoCAD a cuidar automaticamente da operação de cópia incluindo os dados definidos de carregamento.
4.5 – Controle de Visualização

A visualização de entidades no AutoCAD é controlada pela divisão em camadas (Layers). Estas camadas possuem características próprias tais como cor, nome e tipo de linha. Através do comando LAYER do AutoCAD, pode-se alterar as configurações de cada camada (Figura 4.5-1). As três entidades do Modelador foram separadas em camadas onde as subentidades também apresentam uma camada individual.

![Layer & Linetype Properties](image)

Figura 4.5-1 – Quadro de configuração de Layers do AutoCAD.


Finalmente, na entidade **Face**, existem três *layers*. O primeiro, chamado _superfície_, determina a linha de contorno da entidade **Face**. O segundo, chamado _supdir_, determina a representação da direção de trabalho da Superfície. O terceiro, chamado _numsup_, determina a numeração de identificação da entidade **Face**.

Como visto ao longo deste capítulo, as entidades do Modelador apresentam outros dados que possuem representação gráfica. A visualização completa da estrutura implica no controle da exibição dos dados contidos nas entidades de forma organizada. A separação em camadas permite um controle primário da exibição das entidades no que diz respeito apenas às primitivas gráficas. Tomando como exemplo os carregamentos, pode haver situações onde o usuário necessite visualizar somente a geometria das entidades.

Além desta possibilidade, a separação das entidades através de uma faixa de exibição baseada na numeração, pode ser utilizada para visualizar entidades de forma individual ou em grupo.

A implementação do controle de visualização fornece ao usuário uma ferramenta adicional para a visualização das entidades do Modelador dentro do AutoCAD.

A seguir, são descritas a implementação do comando **DISPLAY** do Modelador para controle de visualização e a implementação deste dispositivo em cada tipo de entidade do Modelador.
4.5.1 – Implementação do Controle nas Entidades

Para implementar a funcionalidade do controle de visualização das entidades, foi necessária a utilização de variáveis membros de controle e funções membros de acesso. Foram implementadas em cada entidade três variáveis de controle com exceção da entidade **Node**, que recebeu uma quarta variável de controle com o objetivo de controlar a exibição das restrições nodais. Na Figura 4.5.1-1, estão apresentadas as variáveis e funções para o controle de visualização.

| VARIÁVEIS |
|------------|-----------------|-----------------|
| Nome       | Tipo            | Descrição                                  | Duração   |
| KindCaso   | Adesk::ULInt16  | Código de identificação para a exibição das cargas pertencentes a todos os casos (0), ao caso corrente (1), aos casos habilitados (2) e aos casos desabilitados (4). | Permanente |
| KindLoad   | Adesk::ULInt16  | Código de identificação para a exibição das cargas. (0) para não desenhar nenhuma carga, (1) para desenhar todas as cargas, (2) para desenhar as cargas distribuídas completas, (3) para desenhar as cargas distribuídas parciais, (4) para desenhar as cargas concentradas, (5) para desenhar as opções 2 e 3, (6) para desenhar as opções 2 e 4, (7) para desenhar as opções 3 e 4. | Permanente |
| KindRelease| Adesk::Boolean  | Controle para desenhar ou não as restrições nodais na entidade **Node**. Verdadeiro desenha as restrições, falso não. Esta variável está presente somente na entidade **Node**. | Permanente |
**FUNÇÕES MEMBROS**

<table>
<thead>
<tr>
<th>Nome</th>
<th>Retorno</th>
<th>Objetivos</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>setKindGeom</td>
<td>void</td>
<td>Controla o valor da variável KindGeom.</td>
<td>Específica</td>
</tr>
<tr>
<td>setKindCaso</td>
<td>void</td>
<td>Controla o valor da variável KindCaso.</td>
<td>Específica</td>
</tr>
<tr>
<td>setKindLoad</td>
<td>void</td>
<td>Controla o valor da variável KindLoad.</td>
<td>Específica</td>
</tr>
<tr>
<td>setKindRelease</td>
<td>void</td>
<td>Controla o valor da variável KindRelease.</td>
<td>Específica</td>
</tr>
</tbody>
</table>

Figura 4.5.1-1 – Lista de variáveis e funções membros para controle de visualização.

Estas variáveis permitem que as entidades possam ser desenhadas com base nas opções escolhidas pelo usuário. Em cada entidade, são realizados testes para verificar o valor destas variáveis e desenhar ou não os componentes gráficos de cada entidade.

Como variáveis membros das entidades, estes valores permanecem armazenados nas entidades mesmo quando o arquivo de desenho é gravado. O padrão de visualização das entidades é o desenho completo, geometria e carregamentos, nas três entidades e, adicionalmente na entidade **Node**, a não exibição das restrições nodais.

A alteração destas variáveis implica numa redefinição da exibição da entidade na tela gráfica do AutoCAD. Para que esta redefinição seja garantida, é necessário o acionamento das primitivas de desenho **draw** e **recordGraphicsModified** em cada entidade do Modelador e o comando global **acedUpdateDisplay**. O acionamento desta primitiva permite que as entidades sejam novamente desenhadas após a mudança dos parâmetros de exibição. A execução destas primitivas é efetuada através de uma busca das entidades do Modelador, no banco de dados do AutoCAD, e o acionamento individual das primitivas para cada entidade encontrada. A função utilitária **redrawEnts**
realiza este procedimento recebendo como parâmetro o nome da entidade a ser redesenhada. A estrutura desta função está descrita na Figura 4.5.1-2.

Figura 4.5.1-2 – Estrutura da função `redrawEnts`.

4.5.2 – O Comando DISPLAY

O controle de visualização das entidades do Modelador é feito através do comando DISPLAY. Este comando atua de forma global no desenho do AutoCAD, ou seja, todas as entidades do Modelador presentes no arquivo corrente podem ser alteradas de uma só vez.

As opções para visualização das entidades e dos casos de carregamento são disponibilizadas ao usuário através de um quadro de diálogo. Este quadro, Figura 4.5.2-1, permite que sejam especificados os parâmetros de exibição da acordo com os dados pertencentes a cada uma das
três entidades do Modelador. Todas as informações funcionam de forma conjunta, ou seja, as opções atuam de forma interdependente. Como exemplo, pode ser citada a situação onde se deseja exibir somente as cargas concentradas, dos casos de carregamento habilitados, nos Nós com identificação entre o número 10 e 25; todas as Barras e respectivas cargas com exceção das cargas distribuídas. Nesta caso as opções escolhidas funcionam de forma conjunta e sequencial.

![Display Options](image)

**Figura 4.5.2-1 – Quadro de diálogo do comando DISPLAY.**

Após a seleção das opções no quadro de diálogo, são realizadas as operações de atualização dos valores das variáveis de controle contidas em
cada entidade. Esta atualização é feita separadamente em cada entidade através das funções utilitárias **updateNodes**, **updateBars** e **updateFaces**.

O quadro de diálogo do comando **DISPLAY** apresenta campos de habilitação. O valor destes campos é passado como parâmetro para as funções de atualização. Dentro das funções de atualização são realizadas estruturas de testes destes valores para determinar o valor a ser armazenado nas variáveis de controle das entidades.

Quando o quadro é acionado, as opções selecionadas são as mesmas estabelecidas como padrão, ou seja, desenhar a geometria e carregamento em todas as entidades e não desenhar as restrições nodais.

A possibilidade da utilização de uma faixa de exibição permite que o usuário especifique o valor da numeração inicial e final das entidades a serem exibidas. Se o número inicial for igual ao final somente a entidade que contém este número será exibida. Isto permite que uma entidade seja isolada e exibida de forma completa na tela gráfica do AutoCAD. O controle dos casos de carregamento afetam todas as entidades do Modelador.

Estas possibilidades de exibição, juntamente com o controle de camadas (Layers) do AutoCAD, permitem um controle total da visualização das entidades do Modelador e seus respectivos dados com representação gráfica.

As funções que efetuam o desenho das entidades utilizam os parâmetros para desenhar apenas as opções selecionadas no quadro de diálogo.
4.6 – Exportação de Dados

O objetivo principal desta parte do Modelador é criar uma interface gráfica para o modelamento de estruturas de Barras. Após o modelamento os dados atribuídos às entidades, sejam eles geométricos ou de carregamentos, devem ser exportados para o programa de análise.

Este mecanismo permite a integração do Modelador com vários programas de análise, desde que seja determinado o formato do arquivo a ser utilizado como interface. Uma vez estabelecido o formato e o tipo do arquivo, geralmente um arquivo texto do tipo ASCII, os dados das entidades do Modelador podem ser colhidos e formatados neste arquivo.

A biblioteca ObjectARX utiliza as funções da linguagem C++ para a manipulação de arquivos externos. Estes arquivos podem ter vários formatos e extensões como arquivos do tipo texto, arquivos com formato binário e até arquivos compatíveis com banco de dados.

4.6.1 – Coleta de Dados das Entidades

Todos os dados relativos às entidades estão contidos no banco de dados do AutoCAD. Através da identificação única destas entidades, pode-se acessar os dados, utilizando as funções membros, e copiar estes dados para o arquivo desejado.

Para agilizar o processo de coleta de dados foram implementadas duas funções para construir três vetores contendo as identificações (AcDbObjectId) das entidades a serem exportadas. A primeira função, chamada buildArray, colhe as identificações de todas as entidades do Modelador contidas no banco de dados do AutoCAD. Esta opção significa a exportação completa do arquivo DWG que está sendo utilizado. A segunda função, chamada onPick, permite
que o usuário selecione, através da tela de desenho do AutoCAD, as entidades a serem exportadas de forma individual ou através de uma janela.

Estes três vetores (NodeArray, FaceArray e BarArray) são utilizados como parâmetros para a função de exportação chamada exportData. A função exportData é responsável pela abertura do arquivo de exportação e pela chamada, caso o respectivo vetor contenha entidades, das funções de exportação de cada entidade em separado.

As funções que exportam os dados em cada tipo de entidade são exportNodesAll, exportBarsAll e exportFacesAll. Cada uma destas funções é responsável pela escrita dos dados relativos à entidade respectiva de acordo com os tipos de dados presentes nesta entidade (Figura 4.6.1-1).

Figura 4.6.1-1 – Funções exportNodesAll, exportBarsAll e exportFacesAll.
Os dados são então coletados através dos vetores de identificação das entidades e escritos no arquivo ASCII. O Modelador implementou apenas uma opção de arquivo de exportação no formato ASCII com uma formatação genérica. A partir de implementações futuras, outros formatos de arquivos podem ser implementados com facilidade.

4.6.2 – Arquivo de Exportação

O arquivo de exportação gerado pelo Modelador tem a extensão CAD. Este arquivo está formatado segundo uma divisão criteriosa das propriedades e dados das entidades. As funções descritas no item 4.6.1 utilizam-se da função fprintf, da linguagem C++, para escrever linhas no arquivo ASCII.

A abertura e fechamento do arquivo de exportação é feita através dos comandos fopen e fclose. O acesso ao arquivo é referenciado por um ponteiro C++ para o tipo FILE. Este ponteiro é obtido pela função fopen e fica disponível para as funções de exportação descritas no item 4.6.1.

Para a criação de outros padrões de exportação, as funções do item 4.6.1 podem ser reutilizadas para gerar um formato compatível com programas de análise existentes. A linguagem C++ permite também a manipulação de arquivos de banco de dados e arquivos binários. Em determinados formatos, alguns dados podem ser desnecessários ou dados adicionais, calculados a partir de outros existente, podem ser acrescentados ao arquivo.

A estrutura do arquivo é dividida por entidades e, dentro de cada tipo de entidade é subdividida por dados geométricos e dados de carregamentos. Os dados estão organizados em colunas na formatação adequada. A seguir, é mostrado um exemplo de estrutura modelada no AutoCAD utilizando o Modelador Estrutural (Figura 4.6.2-1).
Figura 4.6.2-1 – Exemplo de estrutura modelada com o Modelador Estrutural.

A estrutura da Figura 4.6.2-1 é composta por 8 Barras dispostas no espaço, 8 Nós e uma Superfície. A Barra 6 apresenta uma carga distribuída completa de -200.0, uma parcial de +500.0 e uma concentrada de -800.0. A Barra 8 apresenta uma carga distribuída completa de -400.0. O Nó 6 apresenta duas cargas concentradas de +400.0(y) e -500.0(x). O Nó 4 apresenta uma carga de +450.0(x). A Superfície 1 apresenta uma carga de -250.0.

O arquivo de exportação no formato ASCII, gerado a partir da estrutura da Figura 4.6.2-1, está descrito a seguir na Figura 4.6.2-2. A divisão do arquivo em seções correspondentes às três entidades pode ser claramente percebida. Após o detalhamento dos dados geométricos estão dispostos os dados relativos a carregamentos em cada tipo de entidade.
<table>
<thead>
<tr>
<th>NOS</th>
<th>Coordenadas</th>
<th>Restrições</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ID</td>
<td>X</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>------</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>2.00</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>300.00</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>3002.00</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>2.00</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>2.00</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>3002.00</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>3002.00</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>2.00</td>
</tr>
</tbody>
</table>

**CARGAS NODAIS**

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Axis</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-500.00</td>
<td>X</td>
</tr>
<tr>
<td>6</td>
<td>400.00</td>
<td>Y</td>
</tr>
<tr>
<td>4</td>
<td>450.00</td>
<td>X</td>
</tr>
</tbody>
</table>

**FACES**

<table>
<thead>
<tr>
<th>ID</th>
<th>DIR</th>
<th>Tipo</th>
<th>NY</th>
<th>ORIGEM</th>
<th>DEMAIS VERTICES</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>90.00</td>
<td>L</td>
<td>4</td>
<td>5</td>
<td>8 7 6</td>
</tr>
</tbody>
</table>

**CARGAS DISTRIBUIDAS SOBRE LAJES**

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Caso</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-250.00</td>
<td>1</td>
</tr>
</tbody>
</table>

**BARRAS**

<table>
<thead>
<tr>
<th>ID</th>
<th>NO1</th>
<th>NO2</th>
<th>Tipo</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>8</td>
<td>P</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>8</td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>8</td>
<td>P</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>7</td>
<td>P</td>
</tr>
</tbody>
</table>

**CARGAS DISTRIBUIDAS UNIFORMES COMPLETAS**

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Caso</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>-400.00</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>-200.00</td>
<td>1</td>
</tr>
</tbody>
</table>

**CARGAS DISTRIBUIDAS UNIFORMES PARciais**

<table>
<thead>
<tr>
<th>ID</th>
<th>a</th>
<th>b</th>
<th>Caso</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>500.00</td>
<td>1000.00</td>
<td>250.00</td>
</tr>
</tbody>
</table>

**CARGAS CONCENTRADAS**

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>b</th>
<th>Caso</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>-800.00</td>
<td>1750.00</td>
<td>1</td>
</tr>
</tbody>
</table>

Figura 4.6.2-2 – Arquivo de exportação no formato ASCII.
4.6.3 – O Comando EXDATA

Para efetuar a exportação dos dados da estrutura modelada ou de parte dela, o Modelador disponibiliza o comando EXDATA. Ao ser acionado, o comando exibe um quadro de diálogo ao usuário (Figura 4.6.3-1).

![Figura 4.6.3-1 – Quadro de diálogo do comando EXDATA.](image)

Ao se acionar o botão BROWSE..., o Modelador utiliza o quadro de diálogo padrão do Windows para possibilitar uma escolha mais cômoda do caminho e nome do arquivo de exportação. A utilização deste quadro padrão traz inúmeras vantagens e simplificações na escolha do nome e caminho do arquivo. Utilizando a biblioteca MFC do Windows (classe CFileDialog), este quadro efetua automaticamente verificações de coerência do nome do arquivo, existência do caminho escolhido, avisos de segurança para evitar cópia de um arquivo sobre outro existente. Neste quadro padrão também é possível navegar pelo disco rígido, criar diretórios, alterar o nome de arquivos, apagar arquivos, etc.
Durante a execução deste quadro, são passados como parâmetros a extensão desejada do arquivo, no caso do Modelador, a extensão `CAD`, o nome padrão do arquivo (**noname.cad**) e outras opções de manipulação de arquivos. A Figura 4.6.3-2 apresenta o quadro de diálculo padrão para salvamento de arquivos do Windows.

**Figura 4.6.3-2 — Quadro de diálogo da classe CFileDialog.**
4.7 – Dispositivos Globais

A estrutura programática do Modelador Estrutural é composta por módulos de bibliotecas dinâmicas (DLL). Estes módulos trabalham em conjunto no ambiente do AutoCAD formando, através deste funcionamento conjunto, o Modelador Estrutural.

Muitos destes módulos apresentam rotinas e arquivos em comum a outros módulos. Além disso, alguns módulos dependem do carregamento prévio de outros módulos para que alguns comandos e operações sejam efetuados com sucesso. Por exemplo, para que uma operação de cópia crie uma nova entidade com numeração atualizada é necessário que o módulo que cria o objeto de numeração já tenha sido carregado.

Este dependência poderia ser implementada em cada módulo de forma individual, ou seja, cada um seria responsável pelo carregamento dos módulos necessários ao seu funcionamento. Neste caso, poderiam ocorrer situações onde um módulo seria carregado duas ou mais vezes, causando assim um erro no AutoCAD.

Cada módulo registra também seus comandos na pilha de comandos (stack) do AutoCAD. Estes comandos permitem o acesso às funções do Modelador Estrutural. Além do acionamento manual, através do prompt do AutoCAD, foram implementadas barras de ferramentas (Toolbars) que permitem o acesso aos comandos do Modelador através de ícones. Estas barras de ferramentas são automaticamente criadas pelo carregamento do Modelador no AutoCAD.

A seguir, são descritas a estrutura do módulo central do Modelador Estrutural e a implementação de barras de ferramentas no ambiente do AutoCAD utilizando a biblioteca MFC do Windows.
4.7.1 – Módulo Central do Modelador

A utilização de um módulo central de controle traz muitas vantagens para o funcionamento e utilização do Modelador. Para controlar o carregamento dos demais módulos, doze no total, foi criado um módulo principal chamado main.arx.

Para se carregar o Modelador no ambiente do AutoCAD, basta que se utilize o comando ARX ou então APPLOAD. Estes comandos permitem o carregamento individual de módulos ARX para o ambiente AutoCAD. Carregando-se o módulo main.arx, ele mesmo se encarrega de carregar os demais módulos do Modelador na ordem adequada.

Se não houvesse um módulo central, o usuário teria que carregar individualmente todos os doze módulos do Modelador em uma ordem certa. Além disso, o descarregamento também deve obedecer um ordem específica para não causar erros no AutoCAD.

Outra vantagem da utilização de um módulo central é a possibilidade da utilização de diversos diretórios para localização dos arquivos. Quando se trabalha com desenvolvimento do projeto no Visual C++, o ideal é que os módulos estejam organizados em Workspaces (ambientes de trabalho). Quando se prevê a utilização do Modelador pelo usuário, o ideal é a organização dos módulos do Modelador em um diretório único de trabalho. Através do módulo central, pode-se então especificar diversos caminhos para busca dos módulos possibilitando que não seja necessária a criação de compilações para desenvolvimento e compilações para utilização.

A estrutura do módulo central de carregamento é relativamente simples e está descrita na Figura 4.7.1-1. Cada módulo é carregado de forma individual através da função loadModule do ObjectARX.
Figura 4.7.1-1 – Módulo central do Modelador Estrutural.
O chamado caminho 1 é o diretório de trabalho do desenvolvimento e, o caminho 2, o diretório padrão do AutoCAD ou outro diretório alternativo para a instalação do Modelador em outra máquina. A ordem de descarregamento dos módulos do Modelador é descrita a seguir e pode ser efetuada através da função `unloadModule` do ObjectARX (Figura 4.7.1-2). O descarregamento independe do caminho pois o AutoCAD somente retira o módulo do seu ambiente de memória e efetua as operações necessárias para tratar das entidades e objetos criados pelo módulo que estará ausente.

![Figura 4.7.1-2 – Ordem de descarregamento dos módulos do Modelador Estrutural.](image)

Ao carregar o módulo central `main.arx`, é apresentado ao usuário um quadro de diálogo de inicialização do Modelador e informações sobre o CADTEC. Para carregar os demais módulos, basta que o usuário acione o
botão LOAD. Os módulos serão carregados e uma barra de progresso será mostrada. Após este carregamento, o Modelador estará pronto para ser utilizado (Figura 4.7.1-3).

Figura 4.7.1-3 – Quadros de diálogo do módulo central do Modelador Estrutural.
4.7.2 – Uso de Barras de Ferramentas

A biblioteca ObjectARX permite, através da MFC (biblioteca do Windows), a implementação de dispositivos gráficos tais como quadros de diálogos, menus, rotinas gráficas e barras de ferramentas. O acesso visual aos comandos do AutoCAD permite uma agilização do acionamento destes comandos de forma bastante simples e intuitiva. A implementação das barras de ferramentas torna a utilização de um aplicativo muito mais fácil e agradável ao usuário.

Os aplicativos desenvolvidos em ObjectARX são executados no ambiente do AutoCAD. Devido a este fato, todos os dispositivos implementados por estes aplicativos devem ser compatíveis com a janela de trabalho do AutoCAD. Assim, a janela do AutoCAD atua como proprietária das outras janelas e componentes gráficos contidos no AutoCAD.

A implementação das barras de ferramentas no Modelador segue os padrões utilizados pelo próprio AutoCAD. As barras de ferramentas podem ser flutuantes ou atracadas em alguma borda da janela de desenho do AutoCAD. Durante a criação destas barras de ferramentas, a localização onde estas serão atracadas pode ser especificada entre as quatro bordas.

Os botões contidos nestas barras de ferramentas apresentam uma representação gráfica (desenho) que deve sempre proporcionar uma fácil associação com o respectivo comando a ser executado. Além disso, é possível a utilização de dicas (tooltips) quando o mouse é colocado sobre os botões.

Quando o módulo toolbars.arx é carregado no ambiente do AutoCAD são criadas, automaticamente, quatro barras de ferramentas atracadas nos quatro cantos da tela de desenho do AutoCAD. Estes quatro grupos contêm os comandos disponíveis no Modelador e estão agrupados de forma lógica e de
acordo com suas funções. Na Figura 4.7.2-1, pode-se observar estas quatro barras de ferramentas e seus respectivos botões.

![Figura 4.7.2-1 – Barras de ferramentas do Modelador Estrutural.](image)

Cada botão aciona um comando do Modelador. Quando o usuário localiza o mouse sobre cada botão, são exibidas as dicas de cada comando associado ao botão.

A seguir, estão descritos os comandos existentes em cada uma das quatro barras de ferramentas assim como as dicas exibidas ao usuário (Figura 4.7.2-2).
1. **GLOBAL**
   - DISPLAY – Entities Display Options
   - RENUM – Apply renumbering over Nodes, Bars and Faces
   - CASOS – Create and manipulate Load CASOS
   - EXDATA – Export drawing data to an ASCII file

2. **NODE**
   - NREL – Control Node release
   - CLNODE – Apply centered loads over Nodes

3. **BAR**
   - LOADS – Apply – Edit – Remove loads over Bar
   - DISLOAD – Apply distributed loads over Bar
   - CENLOAD – Apply centered loads over Bar
   - CHBAR – Change Bar parameters
   - BAR – Create Bars sequentially

4. **FACE**
   - CLFACE – Apply distributed load on Face
   - CHWORK – Change Face work angle
   - CHFACE – Change Face type
   - FACE – Create Face manually
   - FACES – Create Faces automatically

Figura 4.7.2-2 – Descrição dos comandos e dicas de cada barra de ferramenta.
Capítulo 5

MODELADOR ESTRUTURAL

Descrição dos Módulos
Limitações
Utilização
5.1 – Descrição dos Módulos

Conforme detalhado no Capítulo 2, a programação orientada a objetos e a linguagem C++ permitem a organização e a modularização dos programas criados nestes ambientes. No Modelador Estrutural, a divisão do projeto de desenvolvimento em módulos contribuiu significativamente para a viabilização do trabalho em equipe.

Cada módulo objetivou agrupar funções semelhantes do Modelador incidindo sobre cada tipo de entidade. Os doze módulos criados, aliados ao módulo central, formam um conjunto que proporciona a utilização dos comandos, funções e facilidades do Modelador Estrutural dentro do AutoCAD.

Cada um dos módulos contém arquivos exclusivos e utiliza-se de arquivos pertencentes a outros módulos. A descrição dos 13 módulos é a seguinte:

- **basics.arx**: Módulo que cria e instancia o objeto de numeração **Config1** (inicializado com a numeração 1 para todas as três entidades). Implementa os reatores de comandos para operações de cópia e exclusão de entidades. Controla os ajustes e modificações efetuadas após a cópia de entidades.

- **casos.arx**: Módulo que cria e instancia os objetos que controlam os casos de carregamento (**Casos** e **CurrCasos**). Inicia os dois objetos com o caso “DEFAULT” de carregamento. Cria e controla o quadro de diálogo do comando **CASOS**. Controla a remoção dos casos de carregamento nas entidades.

- **clface.arx**: Módulo que implementa o comando **CLFACE**. Implementa também o quadro de diálogo para a entrada das cargas
distribuídas na entidade **Face**. Controla a inclusão dos dados de carregamento na entidade **Face** através de funções membros.

- **clnode.arx**: Módulo que implementa o comando **CLNODE**. Além disso, implementa o quadro de diálogo para entrada de cargas concentradas nas direções X, Y e Z. Controla a inclusão dos dados de carregamento na entidade **Node** utilizando as funções membros.

- **display.arx**: Módulo que implementa o controle de visualização das três entidade do Modelador. Implementa ainda o comando **DISPLAY** e seu quadro de diálogo que permite o controle das opções de exibição de dados e geometria das entidades **Node**, **Bar** e **Face**.

- **export.arx**: Módulo que permite, através do comando **EXDATA**, exportar os dados das entidades do Modelador. Implementa também o quadro de diálogo para seleção das entidades a serem exportadas e escolha do nome e tipo do arquivo de exportação.

- **loadbar.arx**: Módulo que implementa as cargas na entidade **Bar**. Implementa também, através do **prompt** do AutoCAD, os comandos **CENLOAD** e **DISLOAD**. Implementa ainda as rotinas e funções de controle destas cargas na entidade **Bar** [Hüt98].

- **loads.arx**: Módulo que implementa o comando **LOADS** e seu quadro de diálogo para visualização, criação e edição dos três tipos de cargas da entidade **Bar** [Hüt98].

- **main.arx**: Módulo central do Modelador que controla o carregamento e descarregamento sequencial dos demais módulos. Implementa os quadros de diálogo de apresentação do Modelador e do CADTEC.
• nbface.arx: Módulo de criação e implementação das entidades **Node**, **Bar** e **Face**. Implementa também os reatores de entidades. Implementa o procedimento de criação de Barras e Nós de forma automática (comando **BAR**). Implementa e criação e edição da entidade **Face** de forma individual (comando **FACE**) ou automática (comando **FACES**). Disponibiliza o comando **CHBAR** (edição do tipo de Barra), **CHFACE** (edição do tipo de Superfície) e **CHWORK** (edição da direção de trabalho das Superfícies).

• nrelease.arx: Módulo que implementa as opções de liberações na entidade **Node**. Implementa ainda o comando **NREL** e seu quadro de diálogo para edição dos parâmetros de restrições e liberações do Nó quanto à rotação e translação. Controla as funções para atualização das variáveis membros da entidade **Node** relativas às liberações.

• renum.arx: Módulo que implementa a rotina de renumeração. Implementa o comando **RENUM** e seu quadro de diálogo para a escolha dos parâmetros de renumeração. Implementa ainda as rotinas de ordenação em cada uma das três entidades do Modelador.

• toolbar.arx: Módulo que cria e implementa as barras de ferramentas no ambiente do AutoCAD para o acionamento dos comandos do Modelador.
5.2 – Limitações

Apesar das implementações avançadas efetuadas até o presente momento no Modelador Estrutural, existem ainda algumas limitações no funcionamento do programa no contexto estrutural e no contexto do AutoCAD.

Estas limitações motivam a continuação do projeto Modelador. As limitações podem ser descritas da seguinte maneira:

- **Movimentação**: Os comandos que efetuam movimentação de entidades no AutoCAD, tais como MOVE, ROTATE, ALIGN, etc., não efetuam verificações de consistência. Isto significa que, se durante uma operação de movimentação, uma entidade **Node** ficar exatamente sobre outra entidade **Node** o Modelador não efetua uma notificação de erro e nem mesmo corrige o problema. O usuário fica, neste momento, responsável por uma verificação visual deste problema.

- **Fator de Escala**: O Modelador foi inicialmente projetado para trabalhar com estruturas metálicas. Devido a este fato, todas as coordenadas e dimensões estão baseadas na unidade milímetro. Quando as Barras são criadas no AutoCAD, uma unidade de desenho equivale a um milímetro.

- **Exportação de Arquivos DXF**: As entidades não possuem rotinas de gravação de arquivos DXF. Assim, os dados não gráficos das entidades do Modelador não são exportados durante o comando de criação do arquivo DXF no AutoCAD.

- **Criação de Blocos Internos e Externos**: A criação de blocos internos e externos não foi implementada. Assim, os comandos BLOCK e WBLOCK do AutoCAD apresentam resultados imprevisíveis quanto à numeração das entidades e a conectividade de Barras e Nós.
- **Verificações de Cópia de Faces:** Não foram implementadas verificações durante a cópia da entidade **Face**. O Modelador não verifica e corrige, de forma automática, a cópia de **Face** sobre **Face** quando estas são consideradas iguais e com mesma posição espacial.

- **Alteração dos Dados das Camadas (Layers):** O Modelador não permite a alteração dos dados relativos aos **Layers** (camadas) utilizadas nas entidades **Node**, **Bar** e **Face**. Não é possível alterar cor, tipo de linha e nome das camadas.

- **Cargas em Superfícies:** O Modelador implementa somente a carga distribuída na direção Z nas entidades **Face**. Outros tipos de carregamentos como, por exemplo, carga linear e/ou concentrada não foram implementados.

- **Cargas em Barras:** O Modelador implementa somente cargas na direção Z nas Barras. Além disso, não é habilitada a utilização de momentos fletores aplicados nas Barras.

- **Cargas em Nós:** Na entidade **Node**, não é possível a aplicação de momentos.

- **Cópia de Carregamentos:** No Modelador, durante a operação de cópia, não é possível controlar a cópia de carregamentos. Os comandos de cópia efetuam a cópia integral das entidades incluindo os dados de carregamento.
5.3 – Utilização

A utilização do Modelador Estrutural permite o modelamento de estruturas unifilares baseadas em Nós, Barras e Superfícies. Os procedimentos utilizados para a criação de uma estrutura utilizando o Modelador diferem de acordo com as preferências do usuário diante do ambiente do AutoCAD.

O AutoCAD fornece ao usuário comandos que podem efetuar uma mesma tarefa de forma diferente. Esta abertura permite que a utilização do Modelador se torne bastante flexível.

Como exemplo, pode-se citar a criação de uma malha de Barras. Existem diversas opções para a criação desta malha. O usuário pode, por exemplo, criar um quadro padrão e, utilizando o comando ARRAY do AutoCAD, gerar automaticamente o restante da malha especificando o número de linhas, colunas e espaçamentos. Para construir a mesma malha, o usuário pode criar um fila de Barras na direção horizontal e vertical e, utilizando o comando OFFSET do AutoCAD, criar as demais colunas e linhas especificando apenas a distância entre elas.

Outra abertura na utilização do Modelador é a possibilidade de acionar os comandos através do prompt do AutoCAD ou então através das barras de ferramentas implementadas pelo Modelador.

O mecanismo que possibilita a recuperação de comandos (mecanismo do comando UNDO do AutoCAD) pode ser utilizado normalmente sobre as entidades e comandos do Modelador. Assim, quando o usuário efetua uma ou uma sequência de operações erradas, o estado anterior do desenho pode ser recuperado com o acionamento do comando UNDO. Este comando do AutoCAD permite desfazer comandos até a posição em que a sessão do AutoCAD foi iniciada, o arquivo de desenho foi gravado ou um novo desenho foi aberto.
O Modelador torna-se bastante versátil devido à possibilidade de modelar uma estrutura no mesmo ambiente de um projeto arquitetônico. O usuário pode abrir um desenho arquitetônico e, utilizando o Modelador Estrutural, modelar sua estrutura sobre o próprio desenho arquitetônico. Quando o usuário acionar a exportação dos dados estruturais somente as entidades do Modelador serão detectadas e estes dados serão exportados sem qualquer problema.

O usuário experiente no ambiente AutoCAD pode tirar mais proveito ainda dos recursos do Modelador Estrutural. O controle de visualização permite um controle adicional das entidades e o isolamento de Barras ou grupos de Barras para uma melhor visualização. Os recursos 3D da plataforma gráfica do AutoCAD podem ser utilizados normalmente. A estrutura pode ser visualizada de um ângulo aleatório especificado a partir dos comandos do próprio AutoCAD.

Os arquivos do AutoCAD que contêm as entidades do Modelador podem ser passados a usuários que não possuem o Modelador. Este arquivos, quando abertos em computadores que não possuem o Modelador, são tratados de forma especial pelo próprio AutoCAD. O AutoCAD se encarrega de substituir, temporariamente, as entidades não reconhecidas por entidades do tipo Proxy. Estas entidades especiais permitem a visualização gráfica das entidades não reconhecidas mas, não habilitam o usuário a efetuar mudanças nos dados destas entidades. Assim, uma estrutura pode ser modelada e repassada a outros usuários com total segurança. As modificações sobre as entidades, a nível de dados, só podem ser efetuadas por plataformas que contenham o Modelador Estrutural.
Capítulo 6

CONCLUSÕES E PROPOSTAS PARA TRABALHOS FUTUROS

Conclusões
Propostas
6.1 – Conclusões

6.1.1 – Coerência Entre o Modelador e a Engenharia

O desenvolvimento do Modelador Estrutural tomou como base, além dos conceitos modernos de programação, os conceitos da Engenharia Estrutural. O funcionamento do Modelador segue os conceitos básicos de modelamento estrutural assim como mantém a coerência entre as características técnicas e programáticas do aplicativo.

As simplificações utilizadas no Modelador objetivaram a rápida construção de um programa com funcionalidade significativa. Esta estruturação não inviabiliza a continuidade do trabalho e a implementação de novas funcionalidades ao programa. A arquitetura do Modelador permite a implementação de novos módulos e rotinas para tornar seu funcionamento mais versátil e aumentar o nível de automação e “costumização” de projetos de engenharia estrutural.

A utilização de um ambiente gráfico versátil e aberto à programação, o AutoCAD, permite a introdução de conceitos bastante próximos da estrutura real. Esta proximidade proporciona uma interface bastante agradável ao usuário e permite que sua atenção concentre-se nos problemas estruturais e não no funcionamento do programa.

A adaptação do Modelador para outros fins como, por exemplo, para o modelamento utilizando elementos finitos é viabilizada pela estrutura básica do programa onde as entidades e rotinas podem ser reorganizadas para criar grupos de barras com uma forma geométrica qualquer.

Muitas das características do Modelador foram determinadas por aspectos estruturais. Em alguns casos, as simplificações implementadas
tiveram origem em métodos de cálculo e detalhamento a serem acoplados ao funcionamento do Modelador. Estas características restringiram em parte a generalização do Modelador mas contribuíram para a finalização de uma versão inicial do mesmo.

Como já comentado, a programação através da linguagem orientada a objetos deixa o programador bastante a vontade para efetuar mudanças, adaptações e atualizações no aplicativo. A estrutura funcional do Modelador, apesar de não implementar algumas facilidades e dispositivos mais avançados, está totalmente aberta à continuação do projeto.

Certamente, o principal produto obtido no presente trabalho não é o produto por si só e sim a tecnologia da aplicação dos conceitos da computação gráfica na engenharia estrutural.

6.1.2 – Eficiência, Produtividade e Segurança

A utilização do Modelador Estrutural proporcionará significativos aumentos de eficiência, produtividade e segurança em projetos de engenharia estrutural. A utilização de um modelamento gráfico, em substituição a uma entrada de dados numérica, traz inúmeros benefícios ao usuário. Além disso, a possibilidade de copiar e aproveitar estruturas aumenta em muito a eficiência e produtividade nos projetos.

O modelamento gráfico de uma estrutura permite ao usuário detectar problemas e erros de uma maneira muito mais simplificada e imediata. O funcionamento do Modelador limita, em grande parte, alguns erros que poderiam ser cometidos numa entrada numérica. Os efeitos visuais de um comando podem alertar ao usuário a existência de incoerências estruturais e erros de posicionamento da estrutura.
As ferramentas e facilidades do Modelador também proporcionam ao usuário uma tranquilidade maior quanto aos aspectos estruturais inseridos no processo produtivo, tais como numeração, conectividade, coordenadas, etc. A possibilidade de edição de uma estrutura de forma visual também proporciona um ganho significativo de produtividade. Tratando-se de estruturas repetitivas e modularizadas, a eficiência aumenta mais ainda.

Algumas estruturas podem apresentar simetrias que não se baseiam em eixos retos ou então simetrias espelhadas. Com o Modelador, estes tipos de simetrias fora de padrão são tratadas com naturalidade no ambiente do AutoCAD. As ferramentas gráficas disponíveis no AutoCAD tornam o modelamento tridimensional muito mais simples e de visualização facilitada. A geração de estruturas através de operações de cópia também fica bastante simples, mesmo sem que esta tenha qualquer simetria.

A entrada de carregamentos ainda é efetuada através de entrada numérica mas, através de ferramentas gráficas, o Modelador permite uma visualização destes carregamentos sobre a estrutura. Esta visualização permite uma visão global dos carregamentos em uma estrutura assim como a detecção de erros na entrada dos carregamentos.

A coerência entre as entidades e seus carregamentos permite a facilidade de copiar e editar as entidades de forma simplificada. A criação de detalhes estruturais também é facilitada pelo Modelador e pela interface do AutoCAD. Vistas superiores, laterais e inferiores são facilmente obtidas. A utilização de carregamentos através de um módulo que controla os casos de carregamento é bastante simplificada. As cargas são associadas aos casos de carregamento organizando o projeto estrutural.
6.1.3 – Impactos da Computação Gráfica

Os impactos da utilização de um modelador gráfico no sistema CAD são sensíveis e importantes. Seguindo a própria tendência da computação, os aplicativos desenvolvidos para engenharia têm procurado tornar o modelamento de uma estrutura mais amigável e visualmente agradável.

A computação gráfica fornece ao usuário uma ferramenta adicional em apoio à manipulação de um conjunto considerável de dados numéricos. Estes dados, quando transformados em representação gráfica espacial, ressaltam sua importância no contexto estrutural. As influências causadas por uma mudança nestes dados numéricos são detectadas visualmente e em tempo real.

A manutenção de um modelamento estrutural utilizando uma interface visual é uma tarefa mais simplificada. O usuário tem fácil acesso a todo o conjunto de dados e uma interpretação imediata do significado. A base gráfica fornecida pelo modelamento estrutural fornece subsídios para outras aplicações gráficas tais como detalhamento, "renderização" e animações.

A utilização de dispositivos visuais tais como, a interface do sistema operacional Windows e a plataforma CAD do AutoCAD, atuam de forma positiva na motivação e atenção do usuário durante o modelamento de uma estrutura.
6.2 – Propostas

A continuidade do projeto Modelador apresenta novas implementações ao processo de automatização de projetos. A base programática do Modelador fornece ferramentas e conhecimento técnico para a implementação de novas rotinas e funções.

Além do aprimoramento do modelamento estrutural, outras ferramentas podem ser desenvolvidas de forma semelhante. A etapa de detalhamento de projetos, a utilização de modelamento em elementos finitos, a utilização de perfis estruturais em três dimensões são algumas das possibilidades.

O próprio desenvolvimento da plataforma AutoCAD e de sua programação contribuem para implementações mais avançadas e ousadas.

6.2.1 – Criação de Barras Intermediárias

O Modelador implementa somente a criação de Barras partindo e terminando em Nós. Esta característica satisfaz plenamente boa parte das metodologias gerais de modelamento estrutural.

Muitas vezes, a utilização de Barras intermediárias, ou seja, Barras que têm suas extremidades no vão de outras Barras, pode ser necessária. A implementação deste tipo de Barras pode ser inserida no Modelador de forma a viabilizar a entrada de Barras nas duas opções.

Obviamente, a complexidade aumenta significativamente neste caso devido à existência conjunta dos dois tipos de entrada de Barras e devido ao funcionamento das rotinas de verificação de consistência.
A implementação de um modelamento alternativo como este deve ser estudada de forma cuidadosa adaptando-o ao contexto do Modelador. Tal implementação pode exigir algumas mudanças conceituais no funcionamento global do programa e exigir a criação de novas rotinas de verificações de consistência de dados.

6.2.2 – Opções de Escala e Representação Gráfica

O desenvolvimento do Modelador, inicialmente direcionado ao modelamento de estruturas metálicas, utilizou uma base fixa de escala e representação gráfica das entidades.

A utilização prática do Modelador exige a implementação de dispositivos que tornem suas características mais flexíveis e adaptáveis a diversos tipos de usuários.

A implementação de opções de configuração global de escalas proporciona a utilização mais flexível do Modelador para diversos tipos de estruturas, sejam quais forem as escalas de trabalho. Além disso, a representação gráfica de cada entidade pode variar de acordo com a padronização de um profissional ou uma empresa de engenharia.

A estrutura programática do Modelador permite tal implementação a partir da utilização de objetos de configuração e a utilização de rotinas gráficas paramétricas e não, como foi utilizado, baseada em dados fixos. A alteração desta característica não é tão trabalhosa devido à abertura proporcionada pela linguagem de programação utilizada e à arquitetura do programa.
6.2.3 – Interfaces com Programas de Análise

A interface desenvolvida no Modelador Estrutural apresenta uma formatação genérica de um arquivo do tipo texto (ASCII). Esta interface foi desenvolvida com base em objetivos iniciais de comunicação com um aplicativo existente utilizado pela CODEME Engenharia Ltda.

A utilização de outros programas de análise, mesmo que estes utilizem padrões diversificados de formatação de arquivos, pode ser implementada de forma simplificada. A coleta de informações nas entidades do Modelador é efetuada de forma completa e estes dados podem ser formatados da forma desejada.

A possibilidade da utilização de diversas interfaces simultaneamente também é viabilizada pela estrutura do programa. O estrutura da rotina de exportação de dados foi desenvolvida levando-se em conta a possibilidade de existência de diversos modelos de arquivos. A linguagem C++ também torna viável a utilização de arquivos com outras padronizações e formatos diferentes dos arquivos do tipo texto.

A análise de uma estrutura pode determinar mudanças a nível geométrico. O desenvolvimento de uma interface entre o programa de análise e o Modelador pode permitir ainda uma troca bilateral de informações. Esta flexibilidade pode proporcionar um maior desenvolvimento da eficiência do processo.

6.2.4 – Rotinas de Detalhamento

A continuação do projeto Modelador implica certamente na implementação de rotinas de detalhamento. Tomando-se como base o
Modelador Estrutural unifilar e a interface com programas de análise, a base de dados necessária para a criação de detalhamentos de uma estrutura fica disponibilizada. A utilização do ambiente do próprio AutoCAD permite que este processo de automatização do detalhamento seja bastante satisfatório e eficiente.

As rotinas de detalhamento certamente incluem procedimentos teóricos e práticos de engenharia estrutural. Estes conceitos podem ser incluídos na estrutura lógica do programa procurando automatizar ao máximo o processo produtivo.

Estas implementações de detalhamento devem possuir sempre uma interface bastante simplificada e flexível ao usuário. O detalhamento, muito mais do que o modelamento, apresenta uma grande diversidade de opções e características atribuídas aos usuários e empresas.

A integração entre o módulo de modelamento e o de detalhamento deve permitir a existência de um meio bilateral de troca de informações permitindo a edição de dados nos dois módulos sem a perda de coerência. Alguns aspectos são detectados apenas na etapa de detalhamento de um projeto e o retorno e edição dos dados estruturais é freqüente.

O desenvolvimento de um módulo de detalhamento também implica na inclusão de novos dados às entidades existentes no Modelador. A estrutura do programa mantém um nível bastante aceitável para a implementação destes novos dados, assim como a implementação de novos comandos e procedimentos para dar apoio a este novo módulo.

O detalhamento exige a implementação de rotinas gráficas mais avançadas envolvendo a construção de representações gráficas sólidas tridimensionais. Estas rotinas envolvem a criação de procedimentos para tratamento de sólidos, interseção de entidades e projeções no plano e espaço.
Além disso, é necessário o desenvolvimento de um banco de dados com informações sobre perfis estruturais. Esta base de dados é útil durante a execução das rotinas gráficas e permite ainda a criação de procedimentos para cálculo de quantitativos estruturais e até projetões de custos de material estrutural.

Outro ponto bastante complexo na implementação de um módulo de detalhamento é o tratamento de ligações. As ligações apresentam uma grande diversidade de opções e características. Neste caso, a padronização e a criação de modelos é essencial para viabilizar tal implementação.

O desenvolvimento das rotinas de detalhamento proporciona ainda diversos subprodutos para visualização da estrutura de forma tridimensional e volumétrica. Esta visualização permite que o usuário tenha acesso à detecção de problemas muitas vezes observados somente na etapa construtiva da estrutura.

A partir dos dados de detalhamento e modelamento, pode-se implementar ainda a etapa de produção dos elementos estruturais. A utilização de perfis metálicos viabiliza a automação através de máquinas de controle numérico a partir de uma interface adequada. Neste caso, atinge-se o ponto final do processo produtivo de uma estrutura, partindo-se do seu modelamento até sua fabricação.
REFERÊNCIAS BIBLIOGRÁFICAS

Impressa
Internet
On-Line
7.1 – Impressa


7.2 – Internet

WebSITES:

<table>
<thead>
<tr>
<th>Nome</th>
<th>Endereço</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autodesk</td>
<td><a href="http://www.autodesk.com">http://www.autodesk.com</a></td>
</tr>
<tr>
<td>Autodesk Developer Network</td>
<td><a href="http://www.autodesk.com/adn">http://www.autodesk.com/adn</a></td>
</tr>
<tr>
<td>ObjectARX</td>
<td><a href="http://www.autodesk.com/objectarx">http://www.autodesk.com/objectarx</a></td>
</tr>
<tr>
<td>Microsoft</td>
<td><a href="http://www.microsoft.com">http://www.microsoft.com</a></td>
</tr>
<tr>
<td>Visual C++</td>
<td><a href="http://www.microsoft.com/visualc">http://www.microsoft.com/visualc</a></td>
</tr>
<tr>
<td>CADONLINE</td>
<td><a href="http://www.cadonline.com">http://www.cadonline.com</a></td>
</tr>
</tbody>
</table>

FtpSITES:

<table>
<thead>
<tr>
<th>Nome</th>
<th>Endereço</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>ftp://ftp.microsoft.com/</td>
</tr>
</tbody>
</table>

Email:

<table>
<thead>
<tr>
<th>Nome</th>
<th>Endereço</th>
</tr>
</thead>
<tbody>
<tr>
<td>Autodesk Developer Network</td>
<td><a href="mailto:devhelp@autodesk.com">devhelp@autodesk.com</a></td>
</tr>
</tbody>
</table>
7.3 – On-Line

- CD-ROM: Visual Studio 97 Pro – Disco 3 (Visual C++ 5.0)
- CD Collection: Autodesk Developer Network – Agosto / 1997
- CD-ROM: Inside Visual C++ 5.0
- CD-ROM: Developing Professional Applications for Windows 95 and NT Using MFC
- CD-ROM: MFC Developer’s Workshop
- CD-ROM: ObjectARX for AutoCAD Release 14
- CD-ROM: AutoCAD Release 14
Capítulo 8

APÊNDICES

Tipos do ObjectARX
Tutoriais
Versão HTML da Dissertação
Apresentação Multimídia
# 8.1 – Tipos do ObjectARX

## Tipos Globais

<table>
<thead>
<tr>
<th>Tipo</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adesk::Int8</td>
<td>Equivalente ao tipo <code>char</code>.</td>
</tr>
<tr>
<td>Adesk::Int16</td>
<td>Equivalente ao tipo <code>short</code>.</td>
</tr>
<tr>
<td>Adesk::Int32</td>
<td>Equivalente ao tipo <code>long</code>.</td>
</tr>
<tr>
<td>Adesk::UInt8</td>
<td>Equivalente ao tipo <code>unsigned char</code>.</td>
</tr>
<tr>
<td>Adesk::UInt16</td>
<td>Equivalente ao tipo <code>unsigned short</code>.</td>
</tr>
<tr>
<td>Adesk::UInt32</td>
<td>Equivalente ao tipo <code>unsigned long</code>.</td>
</tr>
<tr>
<td>Adesk::Boolean</td>
<td>Equivalente ao tipo <code>int</code>.</td>
</tr>
<tr>
<td>Adesk::kTrue</td>
<td>Valor = 1 – Verdadeiro (<code>TRUE</code>).</td>
</tr>
<tr>
<td>Adesk::kFalse</td>
<td>Valor = 0 – Falso (<code>FALSE</code>).</td>
</tr>
<tr>
<td>AcGeDouble</td>
<td>Equivalente ao tipo <code>double</code>.</td>
</tr>
<tr>
<td>AcDbObject1d</td>
<td>Identificação das entidades no banco de dados.</td>
</tr>
</tbody>
</table>

## Tipos Trigonométricos

<table>
<thead>
<tr>
<th>Tipo</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ads_point</code></td>
<td>Ponto no espaço (obtido da tela gráfica).</td>
</tr>
<tr>
<td>AcGePoint2d</td>
<td>Ponto com coordenadas bidimensionais.</td>
</tr>
<tr>
<td>AcGePoint3d</td>
<td>Ponto com coordenadas tridimensionais.</td>
</tr>
<tr>
<td>AcGeVector2d</td>
<td>Vetor com coordenadas bidimensionais.</td>
</tr>
<tr>
<td>AcGeVector3d</td>
<td>Vetor com coordenadas tridimensionais.</td>
</tr>
<tr>
<td>AcGeMatrix2d</td>
<td>Matriz bidimensional.</td>
</tr>
<tr>
<td>AcGeMatrix3d</td>
<td>Matriz tridimensional.</td>
</tr>
</tbody>
</table>

## Tipos Vetoriais

<table>
<thead>
<tr>
<th>Tipo</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcGePoint2dArray</td>
<td>Vetor de pontos com coordenadas bidimensionais.</td>
</tr>
<tr>
<td>AcGePoint3dArray</td>
<td>Vetor de pontos com coordenadas tridimensionais.</td>
</tr>
<tr>
<td>AcGeVector2dArray</td>
<td>Vetor de vetores com coordenadas bidimensionais.</td>
</tr>
<tr>
<td>AcGeVector3dArray</td>
<td>Vetor de vetores com coordenadas tridimensionais.</td>
</tr>
<tr>
<td>AcGeDoubleArray</td>
<td>Vetor de números do tipo <code>AcGeDouble</code>.</td>
</tr>
<tr>
<td>AcDbIntArray</td>
<td>Vetor de números do tipo <code>int</code>.</td>
</tr>
<tr>
<td>AcDbObjectIdArray</td>
<td>Vetor de identificadores de entidades (<code>AcDbObjectId</code>).</td>
</tr>
</tbody>
</table>

## Entidades e Objetos

<table>
<thead>
<tr>
<th>Tipo</th>
<th>Descrição</th>
</tr>
</thead>
<tbody>
<tr>
<td>AcDbEntity</td>
<td>Classe base das entidades.</td>
</tr>
<tr>
<td>AcDbLine</td>
<td>Linha no espaço.</td>
</tr>
<tr>
<td>AcDbCurve</td>
<td>Curva localizada em um plano.</td>
</tr>
<tr>
<td>AcDbPolyline</td>
<td>Linha formada por segmentos.</td>
</tr>
<tr>
<td>AcDbObject</td>
<td>Classe base dos objetos.</td>
</tr>
</tbody>
</table>
8.2 – Tutoriais

8.2.1 – Grelha

A seguir, são descritos alguns dos passos necessários para a criação de um pórtico plano utilizando o Modelador Estrutural.

**Passo 1**: Utilizando o comando **BAR**, criar duas Barras. Uma horizontal com 3000 mm e outra vertical com 4000 mm.

**Passo 2**: Utilizando o comando **OFFSET**, criar as Barras 3 e 4. Utilizando ainda o comando **OFFSET** ou **COPY** criar a Barra 5 a 6000 mm de distância da Barra 2. Complementar criando as Barras 6 e 7.
Passo 3: Utilizando o comando **MIRROR**, espelhar as Barras criadas no passo 2 em relação ao eixo Y. Utilizando o comando **COPY** copiar a estrutura resultante do comando **MIRROR** criando um módulo igual acima.

Passo 4: Utilizando o comando **FACES**, criar as lajes da estrutura.

Passo 5: Utilizando os comandos **CLNODE** e **DISLOAD**, entrar com os carregamentos nas Barras e Nós.
8.2.2 – Trelilha Plana

A seguir, são descritos alguns dos passos necessários para a criação de uma trelilha plana utilizando o Modelador Estrutural.

**Passo1**: Utilizando o comando BAR, criar duas Barras. Uma horizontal com 2000 mm e outra vertical com 3000 mm.

**Passo2**: Utilizando o comando OFFSET, criar as Barras 3 e 4. Utilizando o comando BAR, criar a Barra 5.
**Passo 3**: Utilizando o comando **COPY**, criar dois módulos a partir do quadro criado no passo 2. Espelhar estes dois módulos.

**Passo 4**: Utilizando os comandos **COPY** e **MIRROR**, complementar a estrutura nas extremidades. Renumear a estrutura com o comando **RENUM**.
**Passo 5**: Utilizando o comando CLNODE, entrar com os carregamentos nos Nós 7, 8, 9, 10, 11 e 12.

### 8.2.3 – Pórtico Espacial

A seguir, são descritos alguns dos passos necessários para a criação de um pórtico espacial utilizando o Modelador Estrutural.

**Passo 1**: Utilizando o comando BAR, criar duas Barras. Uma horizontal com 3000 mm e outra vertical com 3000 mm. Copiar as Barras formando um quadro.
Passo 2: Rotacionar o UCS 90 graus em torno de X. Criar a Barra 5 com 3000 mm de comprimento. Copiar a Barra 5 criando as Barras 6, 7 e 8.

**Passo 4** : Utilizando o comando **COPY**, copiar o módulo do passo 3 para o lado e depois copiar estes dois módulos para cima.

**Passo 5** : Utilizando o comando **FACES**, criar as lajes no primeiro e segundo andares. Carregar as lajes da cobertura utilizando o comando **CLFACE**.
8.3 – Versão HTML da Dissertação

Juntamente com a cópia impressa do presente trabalho é disponibilizada uma versão no formato HTML da Dissertação. Esta versão é compatível com os navegadores Netscape e Internet Explorer a partir de suas versões 3.0. Estes arquivos permitem que o presente trabalho seja observado no ambiente da Internet ou localmente num microcomputador.

A navegação através do trabalho pode ser efetuada através de um menu lateral de ícones no padrão Windows. Para iniciar a navegação, o usuário deve abrir o arquivo principal chamado index.html. Este arquivo carrega automaticamente o menu de navegação e as páginas contidas no presente trabalho.

Devido a conversão de arquivos, algumas imagens serão visualizadas numa qualidade inferior à versão impressa do trabalho. A formatação das páginas também se mostra alterada de acordo com os padrões da linguagem HTML.

8.4 – Apresentação Multimídia

No CD-ROM anexado a esta Dissertação está contida a apresentação em Multimídia (Windows 95/98/NT) utilizada na Defesa do presente trabalho. Para iniciar a apresentação, basta executar o arquivo APRESENTA.EXE contido no diretório raiz do CD-ROM.

Para Internet, basta clicar no Hyperlink abaixo, descompactar o arquivo em qualquer diretório e executar o arquivo APRESENTA.EXE.

DOWNLOAD