UMA ABORDAGEM PARA CRIAÇÃO E COMPARTILHAMENTO DE DADOS DE PEÇAS ATRAVÉS DA INTEGRAÇÃO CAD-RDBMS

ALVINO CESÁRIO DA SILVA JUNIOR

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA MECÂNICA

Florianópolis, 16 Abril de 2001

UMA ABORDAGEM PARA CRIAÇÃO E COMPARTILHAMENTO DE DADOS DE PEÇAS ATRAVÉS DA INTEGRAÇÃO CAD-RDBMS

ALVINO CESÁRIO DA SILVA JUNIOR

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA MECÂNICA COM CONCENTRAÇÃO EM PROJETOS DE SISTEMAS MECÂNICOS, APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA DA UNIVERSIDADE FEDERAL DE SANTA CATARINA

Altamir Dias, D. Sc.
Orientador

Júlio César Passos, Dr.
Coordenador do Curso de Pós-Graduação em Engenharia Mecânica

BANCA EXAMINADORA

André Ogliari, Dr. Eng.

Edison da Rosa, Dr. Eng.

Fernando Antônio Forcellini, Dr. Eng.

João Carlos Espíndola Ferreira, Ph.D.

Florianópolis, 16 de Abril de 2001

"Coragem, coragem, se o que você quer é aquilo que pensa e faz, coragem, coragem, pois sei que você pode mais. Muito mais!"

Trecho da música Por Quem os Sinos Dobram, de Raul Seixas e Carlos Ramussem

Este trabalho é dedicado aos meus pais, à Baía, à minha irmã e à Tainan minha esposa, amiga e companheira.

AGRADECIMENTOS

Agradeço ao professor João Carlos Mendes Carvalho do Departamento de Engenharia Mecânica da Universidade Federal de Uberlândia, pelo fundamental apoio nos meus primeiros trabalhos como pesquisador.

À todo pessoal da Menegotti pelos ótimos momentos no estágio, especialmente ao Henrique, Marildo e Roberto os quais também me apoiaram no desenvolvimento deste trabalho.

Ao professor Fernando Antônio Forcellini, pelas orientações dadas já nos contatos iniciais com a Universidade Federal de Santa Catarina.

Aos grandes amigos e companheiros de trabalho Raimundo, Linhares e professor Altamir Dias, pelas orientações, apoio, discussões e festejados momentos de descontração.

Agradeço também aos amigos(as) Edson, Jeferson, Jairo, Barnabé, Patolino, Ricardinho, Leão, Ronaldo e Rosângela, Rogério e Adriana e todo pessoal de Uruaçu-GO os quais, mesmo sem saber, me trouxeram alegria em vários momentos difíceis.

Agradeço à CAPES pelo apoio financeiro e à Coordenação do Curso de Pós-Graduação de Engenharia Mecânica da UFSC pelo apoio administrativo.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AI: *Artificial Intelligence*

API: *Application Programming Interface*

BD: Banco de dados

B-Rep: *Boundary Representation*

CAD: *Computer Aided Design*

CAD*I: *CAD Interface*

CAE: *Computer Aided Engineering*

CAM: *Computer Aided Manufacturing*

CAPP: *Computer Aided Process Planing*

CGI: *Computer Graphics Interface*

CGM: *Computer Graphics Metafile*

CSCW: *Computer Support Cooperative Work*

CSG: *Constructive Solid Geometric*

DBMS: *Database Management System*

DDE: *Dynamic Data Exchange*

DDL: *Data Definition Language*

DFA: *Design for Assembly*

DFM: *Design for Manufacturing*

DML: *Data Manipulation Language*

DXF: *Data Exchange Format*

EED: *Extended Entity Data*

FTP: *File Transfer Protocol*

GT: Grupos de Tecnologia

HTML: *Hypertext Markup Language*

HTTP: *Hypertext Transfer Protocol*

IGES: *Initial Graphics Exchange Specification*

JDBC: *Java Database Connectivity*

JDK: *Java Development Kit*

JMDL: *Java MicroStation Development Language*

KBE: *Knowledge Based Engineering*

MB: Mega Bytes

MDL: *MicroStation Development Language*

MS: *Microsoft*

NC: *Numerical Control*

ODBC: *Open Database Connectivity*

OLE: *Object Link Embedded*

OODB: *Object Oriented Database*

PC: *Personal Computer*

PDDI: *Product Definition Data Interface*

PDES: *Product Data Exchange using STEP*

PDM: *Product Data Management*

QFD: *Quality Function Deployment*

RAM: *Random Access Memory*

RDBMS: *Relational Database Management System*

SET: *Standard Déchange et de Transfert*

SQL: *Structured Query Language*

STEP: *Standard Exchange of Product Model Data*

TCP/IP: *Transmission Control Protocol / Internet Protocol*

UML: *Unified Modeling Language*

URL: *Uniform Resource Locators*

VDAFS: *Verband der Automobilindustrie Fläncheńschnittstelle*

VRML: *Virtual Reality Modeling Language*

WWW: *World Wide Web*

2D: Bidimensional

3D: Tridimensional

RESUMO

As estratégias atuais de desenvolvimento de produtos compartilham um requisito fundamental: a necessidade de ferramentas computacionais capazes de integrar e coordenar requisitos de projeto durante as atividades de modelamento de produtos. Vários estudos nesta área, apontam para a integração de ferramentas de sistemas CAD comerciais com outras tecnologias de informação. Isso permite combinar e compartilhar de maneira centralizada, dados gerados por sistemas computacionais empregados nas diferentes áreas de conhecimento da empresa.

O presente trabalho segue esta linha de pesquisa. Ele propõe uma abordagem para modelamento de peças mecânicas na fase de detalhamento de projeto, utilizando uma biblioteca com definições de *features* personalizadas. Tais definições são entendidas como unidades informacionais reusáveis de projeto, constituídas de dados geométricos e não geométricos. Elas são representadas através da combinação de recursos de sistemas CAD comerciais de médio ou grande porte, com RDBMS. O objetivo é demonstrar potencialidades e limitações na utilização das ferramentas disponíveis nestes sistemas CAD, para o desenvolvimento de aplicações baseadas na tecnologia de *features*. É apresentada uma revisão do estado da arte em sistemas CAD e RDBMS comerciais, descrição da abordagem proposta, e sua implementação prática.

A abordagem proposta integra informações de engenharia no projeto mecânico para criação de modelos mais completos de peças e também facilita o compartilhamento destes dados. Portanto, ela pode ser usada como base de implementação de um ambiente computacional para Engenharia Simultânea. A reusabilidade de dados possibilita o trabalho com dados de projeto já otimizados, segundo considerações de manufaturabilidade, montabilidade, etc., favorecendo assim a padronização, redução de tempo e custos no desenvolvimento de produtos.

ABSTRACT

The current strategies for product development share a fundamental requirement: the need of computational tools capable of integrating and coordinating design requirements during the product modeling activities. Several researches in this area point to the integration of CAD system tools with other information technologies. It allows combining and sharing data generated by computational systems employed at different knowledge areas of the manufacturing enterprise.

The present work follows this line of researches. It proposes an approach for mechanical parts modeling at the detail design phase, using a personalized feature definitions library. Such definitions are reusable design information units, constituted by geometric and non-geometric data. It is represented by the combination of commercial mid-end and high-end CAD systems resources, with RDBMS (Relational Database Management Systems). The objective is to demonstrate some potential and limitations in the use of available CAD system tools, for feature-based applications development. A review of the state of the art in CAD systems and commercial DBMS are presented, the proposed approach and its practical implementation are described.

The proposed approach integrates engineering information in mechanical design, for creating more complete product models. It also facilitates product data sharing, so that it can be used as a Concurrent Engineering computational environment. The data reusability provided enables the work with optimized design data, according to manufaturability aspects, assemblability, etc., favoring the standardization and reduction of product development time and costs.

1 CLARIFICAÇÃO DA TAREFA

1.1 INTRODUÇÃO

Com a crescente competitividade entre as empresas, a redução de tempo e custos do processo de desenvolvimento do produto tornam-se questões cruciais. Na busca de tais objetivos, tem sido implementadas novas estratégias de desenvolvimento do produto. Estas estratégias compartilham um requisito fundamental: a necessidade por ferramentas computacionais capazes de integrar e coordenar as atividades realizadas para o modelamento do produto (KRAUSE et al., 1993). Este requisito tem conduzido e forçado a evolução tecnológica dos sistemas CAD.

Deste modo, para otimizar o emprego destes sistemas, são necessárias abordagens introduzidas no contexto de integração de conhecimentos proposto pelas novas estratégias de desenvolvimento do produto. Este é o campo de estudo do presente trabalho. Os três próximos itens objetivam investigar e estabelecer uma visão geral do assunto. São abordados o processo de desenvolvimento e modelamento do produto e descritos alguns trabalhos referenciais. Pretende-se com isso fundamentar as motivações para a proposta de trabalho e argumentação inicial necessária para a descrição dos objetivos e contribuições da mesma.

1.2 PROCESSO DE DESENVOLVIMENTO DO PRODUTO

Segundo CLARK e FUJIMOTO (1991): "o desenvolvimento do produto é o processo pelo qual uma organização transforma dados sobre oportunidades de mercado e possibilidades técnicas, em bens e informações para a fabricação de um produto comercial". Ele pode ser caracterizado por um ciclo composto de algumas atividades que vão desde a identificação das necessidades do mercado, até a produção piloto ou construção de um protótipo do produto projetado (SILVA e ALLIPRANDINI, 2000).

O desempenho do processo de desenvolvimento do produto pode ser avaliado por três parâmetros básicos: qualidade, tempo e produtividade. "Estes parâmetros devem ser otimizados para capacitar uma empresa na sua habilidade de atrair e satisfazer seus clientes, aumentando a competitividade de seus produtos" (VALERI e ROZENFELD, 2000). O ponto de partida para alcançar esta meta é a atividade de projeto, durante a qual devem

ser considerados os aspectos da boa qualidade do produto em relação ao seu ciclo de vida como um todo (BACK e FORCELLINI, 1999).

Um aspecto importante para o bom andamento do processo de desenvolvimento do produto diz respeito à consistência dos dados gerados nas atividades de projeto. Segundo BACK, N. e FORCELLINI, F. A. (1999), para que o desenvolvimento de produtos se torne efetivo e eficiente, o processo de projeto precisa ser planejado cuidadosamente e executado sistematicamente. O procedimento sistemático deve ser capaz de integrar e otimizar os diferentes aspectos envolvidos no projeto, se adequando às várias tecnologias e possibilitando a interação entre o pessoal envolvido, de modo que todo o processo seja lógico e compreensível.

Existem várias propostas de descrição formal do processo de projeto. Nelas, o projeto como um todo é dividido em etapas seqüenciais usualmente denominadas de (CUNHA e DIAS, 2000), (OGLIARI, 1999), (PAHL e BEITZ, 1995):

- Projeto informacional ou clarificação da tarefa: Nesta etapa lida-se com informações de mercado, transformando-as em especificações de projeto que caracterizam os problemas técnicos e as restrições a serem resolvidas.
- Projeto conceitual: As especificações de projeto são quantificadas em termos de funcionalidades e princípios de solução e representadas através de esquemas ou esboços das formas aproximadas do produto.
- Projeto preliminar: As funções do produto e princípios de solução são transformadas em um leiaute do produto, representado pelos elementos que o constituem e seus parâmetros geométricos principais.
- Projeto detalhado: Nesta etapa ocorre o detalhamento da configuração do produto.

Naturalmente, estas diferentes etapas não são necessariamente seqüenciais, mas envolvem interações e decisões tomadas simultaneamente. De maneira geral, inicia-se o projeto com a identificação de demandas de mercado, especificação dos requisitos para o novo produto e documentação das mesmas num modelo resumido. Este modelo é, então, progressivamente analisado e avaliado, modificado e refinado, buscando as melhores soluções segundo propósitos previamente identificados, até que esteja completamente definido para manufatura.

Esta linha geral de trabalho ainda pode variar entre dois tipos básicos de projetos: o inovador, onde um produto novo é criado e o de evolução, quando o que se cria são variações de produtos pré-existentes. No inovador, as etapas iniciais do processo de projeto são de fundamental importância e devem ser executadas com maior critério. No evolutivo, as etapas iniciais de projeto não são tão importantes e o aproveitamento de trabalhos anteriores é incrementado.

Analisando as atividades de projeto, verifica-se que o processo de desenvolvimento do produto se baseia no uso intensivo de conhecimentos técnicos, mercadológicos, etc. Portanto, deve ser levado em conta a necessidade de trabalho cooperativo e a comunicação efetiva entre áreas de conhecimento distintos, presentes no sistema de manufatura. "O aproveitamento efetivo de soluções de manufatura e engenharia já existentes é vital" (SHAH e MÄNTYLÄ, 1995). Estes fatos foram motivações para o desenvolvimento de várias estratégias melhoradas para o desenvolvimento do produto, tal qual a Engenharia Simultânea.

A Engenharia Simultânea é hoje amplamente reconhecida como um dos meios mais eficientes para o aumento da competitividade de uma empresa. Nesta "filosofia de trabalho", busca-se aumentar a eficácia do produto através da incorporação de todo conhecimento relevante nos estágios iniciais de projeto, e melhorar a eficiência do processo de projeto, pela redução do tempo de desenvolvimento do produto, alcançando objetivos tais como (CHANG et al., 1998):

- Satisfação das necessidades dos clientes (alta qualidade e baixo custo).
- Menor chance de reprojetos.
- Melhor uso dos recursos disponíveis na empresa.
- Capacidade de trabalhar com maior variedade de produtos, etc.

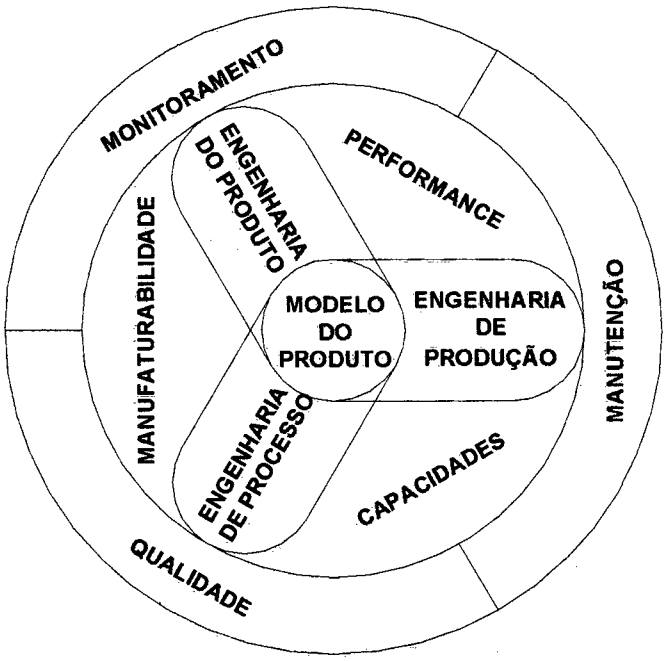


Figura 1.1: O modelo de três pontas da Engenharia Simultânea (CHANG et al., 1998)

Esta filosofia é representada por CHANG, T-. C. et. al. (1998), como uma roda de três raios: Engenharia de Produto, Processo e Produção (Figura 1.1). Conforme o autor,

"... a Engenharia de Produto define características e atributos do produto, as quais junto à base de conhecimentos de processo ditam o quanto manufaturável é o produto. A interface Engenharia de Processo e Produção define a capacidade do sistema de manufatura. A interface Engenharia de Produto e Produção, determina a performance do produto no mercado...".

A implementação prática da Engenharia Simultânea deve considerar, além do sólido relacionamento com clientes e fornecedores, o embasamento em equipes de trabalho multifuncionais e o auxílio computacional. A aplicação de técnicas para formalizar os processos de geração, arquivamento e troca de informações de maneira integrada toma-se a principal preocupação. O modelo do produto passa a ocupar uma posição central no seu ciclo de desenvolvimento, conforme ilustrado na Figura 1.1 (SILVA Jr. e DIAS, 2000).

1.3 MODELAMENTO DO PRODUTO

Os modelos do produto são vitais, contudo, representam apenas uma das fontes de informação necessárias para conseguir a integração completa em um sistema de manufatura. A atividade de "modelamento do produto" trabalha com um conceito mais amplo. Ela diz respeito à integração de dados do produto, com o objetivo de criar um modelo que atenda às necessidades das atividades presentes em todas as etapas do seu ciclo de vida. Segundo KRAUSE et al. (1993), o termo "modelamento do produto" está relacionado com dois tópicos: "modelos do produto" e "fluxo das informações no desenvolvimento do produto".

Modelos do produto são agrupamentos lógicos de todas as informações relevantes relacionadas ao produto criadas durante seu ciclo de vida. No âmbito computacional, eles armazenam informações na forma de dados digitais e são equipados com algoritmos de acesso e manipulação. O fluxo das informações no desenvolvimento do produto representa o processo de modelamento do produto, constituindo-se de um grupo de funções técnicas e gerenciais, requeridas para transformar idéias iniciais em produtos prontos para serem colocados no mercado (KRAUSE et al., 1993). Estas funções técnicas e gerenciais podem ser representadas em termos de vários outros modelos que tem sido denominados de forma variada na literatura. A Figura 1.2 mostra a conexão de modelos relacionados com o produto segundo McMAHON, C. e BROWNE, J. (1998).

Portanto, os sistemas computacionais apropriados para o modelamento do produto devem estar aptos a suportar o fluxo das informações no desenvolvimento do mesmo e armazenar estes dados no modelo do produto. Estudos nesta área identificam a necessidade de formatos padrões para descrição completa do produto e implementação

prática de sistemas computacionais integrados às várias atividades presentes no sistema de manufatura.

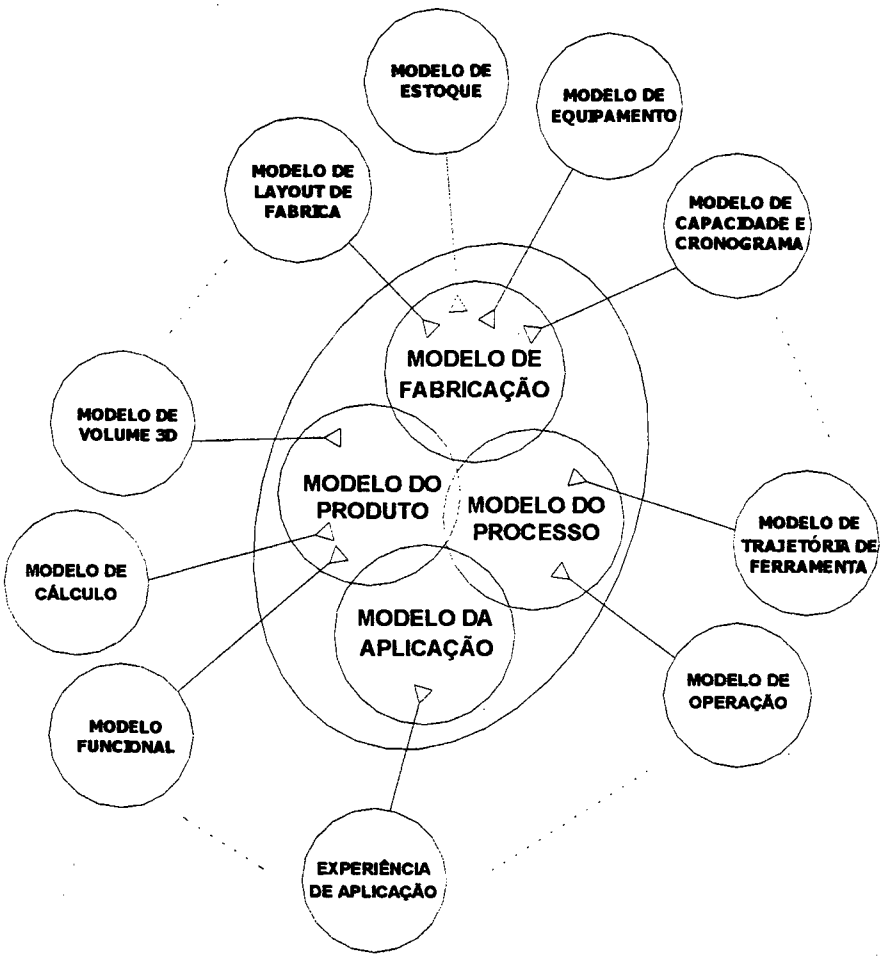


Figura 1.2: Conexão de modelos relacionados com o produto
(McMAHON e BROWNE, 1998)

Os formatos padrões vêm evoluindo ao longo dos anos. A partir dos formatos baseados em geometria: IGES, SET, etc. (McMAHON e BROWNE, 1998), desenvolveu-se outros mais completos: PDDI, PDES, STEP, etc. (McMAHON e BROWNE, 1998), (SSEMAKULA e SATSANGI, 1990), (AN et al., 1995), (KERN, BOHN e BARCIA, 1996) e (KERN e BOHN, 1995). Contudo, também figuram nesta área de estudo outras iniciativas para criação de modelos de dados do produto com propósitos mais específicos, dentre as quais podemos citar: (LINHARES, 2000), (LIMA, 1994), (MAZIERO, 1998) (WANG e WALKER, 1989), (BHAT e BEAGAN, 1989) e (ROSA et al., 1992).

Com respeito à implementação dos sistemas computacionais, ela pode ser executada com diferentes estratégias. Uma abordagem interessante, devido ao caráter urgente das atuais demandas industriais e que pode representar um passo em direção ao

modelamento completo do produto, é a combinação ou integração das potencialidades de ferramentas CAD com outras tecnologias de informação presentes no mercado, tais como: sistemas CAE, CAPP, CAM, PDM, RDBMS, KBE, etc. (KRAUSE et al., 1993).

Existem várias dificuldades associadas à integração de sistemas computacionais devido à complexidade envolvida e à natureza distinta dos dados presentes em cada um deles. Por exemplo, sistemas CAD possuem, na maioria das vezes, uma base de dados centrada em informações geométricas e topológicas de baixo nível semântico. Sistemas CAM por sua vez, lidam com informações de manufatura de alto nível representadas em termos de dimensões da peça, tolerâncias, processos de manufatura, acabamentos superficiais, etc. Para integração de dados entre estes dois sistemas, sem intervenção humana, é preciso estabelecer padrões comuns de comunicação. Alguns trabalhos sugerem que isso pode ser feito através do encapsulamento das informações relevantes para cada sistema em hierarquias de dados abstratos, utilizando técnicas de programação orientada a objetos (MOTAVALLI et al., 1997).

O uso de técnicas de programação orientada a objeto para melhorar as as capacidades de modelamento de sistemas CAD e automatizar sua integração com outros sistemas tem sido desenvolvida ao longo dos anos. Vários estudos neste sentido em atividades de engenharia mecânica vêm sendo baseados na tecnologia de *features*[1]. Um número crescente de propostas tem empregado esta tecnologia para auxílio ao projeto mecânico, por ela tornar possível o encapsulamento de dados semânticos de engenharia, geométricos e não geométricos, numa forma computável. Atualmente, sistemas CAD comerciais de médio e grande porte apresentam ferramentas baseadas no conceito de *features*, as quais devem ser exploradas em todas suas capacidades para otimizar o modelamento do produto.

1.4 TRABALHOS REFERENCIAIS

Vários pesquisadores têm discutido sobre a integração de informações de engenharia no projeto mecânico, através do desenvolvimento de ambientes computacionais que suportem várias ferramentas de auxílio ao projeto.

EYNARD, B. et al. (2000) por exemplo, trabalharam na proposição de métodos e ferramentas para melhorar o projeto de sistemas mecânicos. Eles fizeram uma síntese de vários trabalhos anteriores relacionados com modelamento do produto, otimização de sistemas mecânicos e com a contribuição dos conhecimentos de manufatura no projeto. Eles apresentaram um modelo de produto capaz de formalizar o conhecimento gerado

[1] Neste trabalho, os termos de línguas estrangeiras são escritos em itálico.

durante todo processo de desenvolvimento do produto. O modelo é baseado em conceitos de função, *features* (convencionalmente chamadas de entidades tecnológicas) e entidades de restrição. Acredita-se que com sua implementação em um sistema CAD, poderão ser integrados vários mecanismos de auxílio ao processo de desenvolvimento do produto. Estes estudos identificaram que, para implementar a proposta, o problema da captura de dados e restituição de propriedades informacionais são aspectos fundamentais.

KRISHNASWAMY, G. M e ELSHENNAWY, A. K. (1993), descrevem um ambiente computacional para implementação da engenharia simultânea baseado no uso de sistemas CAD (AutoCAD da Autodesk) e RDBMS (Access da Microsoft). A equipe de desenvolvimento do produto ajuda o cliente a comunicar suas necessidades através de desenhos CAD. Tais desenhos são criados com *features* ou objetos paramétricos, pré definidos em termos de capacidades de manufatura da empresa. O RDBMS é usado como um sistema de gerenciamento dos modelos do produto para manipular e interagir com atributos dos desenhos. Ele também relaciona relatórios de pesquisas de mercado, informações sobre processos de manufatura disponíveis, algoritmos de análise de projeto e detalhes de vendas. Um sistema especialista comunica-se com esta base de dados, e usa regras heurísticas de DFM e DFA para sugerir alterações inteligentes no projeto. A gerência de dados de engenharia é um requisito técnico fundamental para implementação da engenharia simultânea. Comandos computacionais (macros) com acesso a dados de várias aplicações podem conduzir ao sucesso de um ambiente de manufatura integrado.

OZAWA, M. et al. (1998) por sua vez, propõem uma abordagem para acelerar o processo de projeto pela redução de atrasos associados com a troca de informações. Sua abordagem é fundamentada no uso de uma mistura de programas autônomos e interfaces de software (denominados "agentes"), através das quais os projetistas trocam informações entre ferramentas usadas nos vários níveis de representação de conhecimento, durante o processo de projeto. A Tabela 1.1 mostra estes vários níveis e as ferramentas relacionadas. São definidos os conceitos de uma Linguagem de Modelamento Composicional (CML), na qual modelos podem ser criados seguindo algumas convenções e restrições em um Ambiente Colaborativo de Modelamento de Dispositivos (CDME). O trabalho foi concentrado na apresentação de métodos para compartilhar as informações entre o nível (d) (ver Tabela 1.1), com a simulação dinâmica de um produto feito no MatLab e nível (e), com modelos geométricos parametrizados criados no sistema CAD I-DEAS. Deste modo, através de um estudo de caso, foi conseguida a descrição completa da dinâmica de um produto. Contudo, uma série de considerações tiveram que ser estabelecidas manualmente. Com a implementação de agentes que compartilhem informações entre os demais níveis de representação de conhecimentos, será possível uma maior automação do processo.

Tabela 1.1: Classificação dos nívels de representação do conhecimento no projeto. Adaptado de (OZAWA et al., 1998).

ETAPA DO PROCESSO DE PROJETO	NÍVEL DE REPRESENTAÇÃO DE CONHECIMENTO	FERRAMENTAS OU RECURSOS
Clarificação da tarefa e projeto conceitual	(a) Modelo conceitual: Requisitos de mercado e especificações de projeto	QFD, TRIZ
Projeto conceitual e preliminar	(b) Generalidades, Fragmentos do modelo e grupo de restrições	Modelos CML
Projeto preliminar	(c) Equações matemáticas: Modelos compartilháveis intermediários	Mathematica, Planilhas de projeto, CDME
Projeto preliminar e detalhado	(d) Parâmetros: Modelos compartilháveis de baixo nível	MatLab, 3D-CAD (I-DEAS), CDME
Projeto detalhado	(e) Modelo do produto: Modelos CAE 3D ou FEM	I-DEAS (CAD), Pro-Engineer (CAD), STEP, etc.

DONG, A. e AGOGINO, A. M. (1998), descrevem uma abordagem para gerenciar informações de projeto baseadas em sistemas CAD, num ambiente de projeto distribuído através de redes de computadores. Foi caracterizado o desenvolvimento de uma base de conhecimentos compartilhada para troca de informações de produto entre equipes de projeto multi-disciplinares. A base de conhecimento é formada pela associação de modelos geométricos com documentações relacionadas, tais como: memorandos, planilhas e resultados analíticos, etc. O sistema usa uma base de dados relacional e um sistema de gerenciamento de documentos para interligar dados de projeto e para coordenar a execução de ferramentas de projeto. A combinação de base de dados relacional e tecnologia de gerenciamento de documentação, possibilita a utilização das informações baseadas em sistemas CAD como uma interface única para as ferramentas de projeto, e para manipular os dados de projeto.

Outros trabalhos neste sentido mostram ambientes computacionais que completam as funcionalidades de sistemas CAD comerciais, através de sua integração com uma base de dados não geométricos, utilizando a tecnologia de *features*.

WANG, M. e WALKER, H. (1989) por exemplo, formalizaram o conhecimento para o planejamento de processos de usinagem, identificando algumas regras. Então, apresentaram uma abordagem para criação de um "Sistema Inteligente de Planejamento de Processo" com um RDBMS. O sistema aceita a descrição de uma peça em termos de *features* e gera um esboço do seu processo de manufatura. Cada operação é então detalhada pelo relacionamento das *features* aos processos, ferramentas e máquinas, em seguida são acionados algoritmos de otimização para encadear as etapas dos processos identificados, numa seqüência lógica. Dentre os vários aspectos que necessitam melhor

desenvolvimento neste sistema, está a necessidade de disponibilizar uma biblioteca de *features* no estágio de projeto utilizando um sistema CAD. Deste modo as peças modeladas podem alimentar o Sistema Inteligente de Planejamento de Processo automaticamente, tornando possível a implementação de algoritmos que utilizem os dados do sistema para otimizar o projeto em termos das capacidades de manufatura da empresa.

BHAT, S. K. e BEAGAN, D. D. (1989) desenvolveram na GM (General Motors) um trabalho semelhante ao anterior. Através de uma base de dados de *features* denominada "Engine Technology System" (ETS), eles disponibilizam informações atualizadas sobre tecnologias e processos disponíveis, para todos os engenheiros responsáveis pelo projeto e manufatura de componentes de motores. O ETS consiste de blocos de informação associados. Os blocos são peças e suas *features*, as associações relacionam as "montagens" peça-*features*, conjunto-peças. A representação dos dados em termos de *features* disponibiliza um meio bastante intuitivo de manipulação dos mesmos, mas ainda é necessário aperfeiçoar as interfaces gráficas para tal. Os blocos de informação poderiam ser incluídos automaticamente na base de dados do ETS se as *features* fossem disponibilizadas para modelamento de peças num ambiente CAD.

ROSA, E. et al. (1992), propuseram um sistema de modelamento de produtos com o objetivo de desenvolver um ambiente computacional que viabilize a implementação de conceitos de Engenharia Simultânea. A proposta tem como requisito uma base de dados que permita a derivação de diferentes visões do produto (multi-modelamento), durante seu processo de desenvolvimento. Para tal, o trabalho apresenta detalhes de uma estrutura de dados baseada na tecnologia de *features*. Cada *feature* se relaciona com informações relevantes sobre os produtos. Estes dados são divididos em termos de parâmetros numéricos, atributos textuais e índices resultantes de cálculos adicionais. "Tendo este conjunto de informações alocado a nível de *feature* e armazenado em um banco de dados, torna-se factível estabelecer estratégias de processamento destas informações por diferentes aplicativos, sendo este o ponto de partida para o multi-modelamento". Para implementação do sistema é proposta a utilização de um sistema CAD comercial. Nele serão agregadas as regras de inserção das *features* e para ligação com banco de dados não gráfico, no qual são armazenados parâmetros, atributos e índices associados a cada *feature*. O uso de sistemas CAD como núcleo sobre o qual diferentes aplicativos são desenvolvidos garante a portabilidade do sistema, consistência dos dados, e também permite agilizar todo o processo de desenvolvimento do produto.

1.5 DESCRIÇÃO GERAL DA PROPOSTA DO TRABALHO

O presente trabalho segue a mesma linha dos estudos descritos no item anterior, concentrando-se nas questões relacionadas à integração de sistemas computacionais para o modelamento de produtos. Ele propõe uma abordagem que integra informações de engenharia no projeto mecânico, representadas através da combinação de recursos de sistemas CAD comerciais de médio ou grande porte com RDBMS, utilizando a tecnologia de *features*. A proposta consiste em criar uma biblioteca de informações geométricas e não geométricas de projeto, estruturadas em termos de definições personalizadas de *features* e integradas ao sistema CAD, de modo que sejam reusáveis nas atividades de detalhamento de projetos mecânicos.

A aplicação prática desta proposta depende da implementação de recursos computacionais que permitam o modelamento de produtos utilizando as definições personalizadas de *features* de maneira automática e intuitiva. Também é necessária a determinação de uma linha ou classe de produtos específicos para a criação da biblioteca de informações reusáveis de projeto.

Os aspectos relacionados à reusabilidade de dados são entendidos como uma forma de melhorar a qualidade do modelamento do produto. Eles possibilitam o trabalho com dados de projeto já otimizados segundo considerações de manufaturabilidade, montabilidade, etc., favorecendo assim a padronização, redução de tempo e custos.

1.6 OBJETIVOS E CONTRIBUIÇÕES

O objetivo global do presente trabalho é realizar um estudo que permita visualizar as potencialidades e limitações de utilização dos recursos disponíveis em sistemas CAD comerciais, para o desenvolvimento de aplicações baseadas na tecnologia de *features*. Para tal, tem-se como objetivo específico, a implementação de funcionalidades básicas de uma ferramenta que possibilite esta visualização, empregando a abordagem de integração CAD-RDBMS, descrita em linhas gerais no item anterior.

Através dos estudos necessários para a realização deste trabalho, é estabelecida uma base de conhecimentos sobre as tecnologias presentes nos sistemas CAD atuais e tendências para os novos desenvolvimentos. Uma visão clara destes aspectos é um importante ponto de partida para trabalhos futuros nesta área. Isto pode evitar a recriação de soluções já existentes, uso inadequado de recursos e confusões conceituais que prejudicam o entendimento do assunto como um todo.

A implementação prática da abordagem proposta permite a criação de modelos de peças mecânicas mais completos, uma vez que ela integra informações comumente

tratadas por sistemas CAD com outras registradas no RDBMS. A construção destes modelos também é agilizada, já que o projetista é conduzido à trabalhar com entidades de modelamento personalizadas, muito mais intuitivas. Além disso, a disponibilização das informações sobre as *features* usadas no modelamento de um produto em um banco de dados relacional, possibilita o fácil compartilhamento de informações sobre o produto modelado. Este banco de dados pode ser acessado e processado por outras aplicações, úteis em diferentes áreas presentes no ciclo de desenvolvimento do produto.

1.7 EXPOSIÇÃO DO TEXTO

O conteúdo dos próximos capítulos pode ser dividido em três partes. A primeira parte é constituída pelos capítulos 2, 3 e 4, e contém a revisão bibliográfica, que descreve conceitos fundamentais para o entendimento deste trabalho. No Capítulo 2 procura-se estabelecer o estado da arte dos sistemas CAD, abordando suas características genéricas, limitações e técnicas empregadas para melhoria de sua capacidade de modelamento. No Capítulo 3, faz-se um estudo mais aprofundado sobre o modelamento baseado em *features*, uma tecnologia que tem sido reconhecida como o próximo passo para modelamento de produtos com sistemas CAD. No Capítulo 4 é feita uma descrição de conceitos básicos relacionados aos sistemas de gerenciamento de banco de dados, especialmente os RDBMS.

A segunda parte contém o desenvolvimento prático deste trabalho, sendo composta pelos capítulos 5 e 6. No Capítulo 5, a proposta para exploração da tecnologia de *features* disponibilizada em sistemas CAD comerciais, já apresentada no item 1.5, é descrita detalhadamente. O Capítulo 6 descreve os resultados práticos alcançados com a implementação da proposta utilizando o sistema CAD MicroStation/J Modeler da Bentley e o RDBMS Access 97 da Microsoft.

A terceira e última parte, Capítulo 7, contém a análise, discussão dos resultados e considerações finais.

2 CARACTERÍSTICAS GERAIS DE SISTEMAS CAD COMERCIAIS

2.1 INTRODUÇÃO

Os sistemas CAD podem ser entendidos como uma coleção de aplicativos que processam os dados armazenados na base de dados de diferentes formas, utilizando as funções de acesso suportadas. A visão de uma arquitetura centralizada, conforme mostra a Figura 2.1, é um modo razoável para compreender esta idéia (STALEY e ANDERSON, 1986).

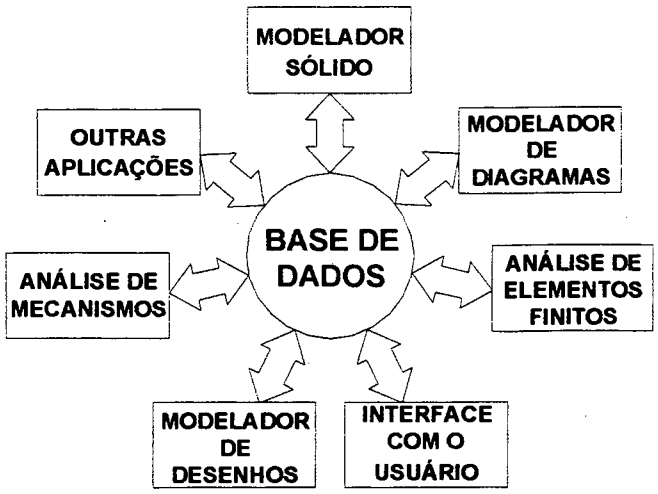


Figura 2.1: Sistema CAD com arquitetura centralizada na base de dados (STALEY e ANDERSON, 1986)

Desde os primórdios de seu desenvolvimento, fim da década de 50 e início da de 60, os sistemas CAD têm como objetivo fundamental acelerar as atividades do ciclo de produção, melhorar a qualidade, e aumentar a produtividade do projeto mecânico e produtivo através da integração e automação das atividades de projeto. Inicialmente estes sistemas possuíam ferramentas de desenho que permitiam a criação de modelos 2D, tornando-se assim, pranchetas eletrônicas. Atualmente, com hardware e software melhores e mais baratos, a maioria dos sistemas CAD fazem uso intensivo de técnicas de modelamento 3D, disponibilizam grande facilidade de manipulação, análises, simulação e visualização gráfica da forma e estrutura do produto, além de contarem com meios mais eficientes para trabalho em rede (McMAHON e BROWNE, 1998), (ZEID, 1991).

Contudo, a base de dados de sistemas CAD ainda é centralizada em entidades gráficas, as quais, apesar de extremamente importantes, satisfazem apenas uma parte dos requisitos para a documentação do projeto. Sabe-se que os projetistas estão interessados em informações que não estão em documentos de projeto constituídos somente de desenhos e especificações. Os novos desenvolvimentos de sistemas CAD objetivam atuar em ambientes altamente integrados, lidando com vários tipos de informações, vindas de diferentes áreas de conhecimento do sistema de manufatura (KUFFNER e ULLMAN, 1991), (McMAHON e BROWNE, 1998).

Neste capítulo, o contexto atual de desenvolvimento de sistemas CAD é investigado em linhas gerais, buscando esclarecer o estado da arte. São estudados os métodos de representação, armazenamento, recuperação e troca de informações de projeto com sistemas CAD. Em seguida são apresentadas algumas técnicas que têm sido empregadas em novos desenvolvimentos nesta área.

2.2 REPRESENTAÇÕES DO PROJETO EM SISTEMAS CAD

Freqüentemente é exigido que o projetista tome decisões com base em propriedades morfológicas do produto. Por este motivo o desenho geométrico é a mídia mais usada para comunicação nas atividades de engenharia. A geração destes desenhos é a principal função dos sistemas CAD nas empresas. Isto é feito utilizando técnicas de modelamento 2D ou 3D.

2.2.1 Modelos 2D

No ambiente de projeto, a complexidade envolvida aliada à necessidade de evitar interpretações ambíguas indica que os modelos devem estar em conformidade com padrões amplamente reconhecidos. Tradicionalmente, isto é feito com a utilização de desenhos de componentes e diagramas.

Os desenhos, Figura 2.2, são elaborados em conformidade com os conceitos da geometria descritiva. Ela consiste de técnicas para representação de formas 3D em espaços 2D, através da projeção paralela de vistas do objeto em planos mutuamente perpendiculares denominada de Projeção Ortográfica. Nos diagramas, Figura 2.3, a estrutura lógica ou física de um sistema, em termos da montagem das partes primitivas e relacionamento entre elas, é mostrado por uma série de símbolos e conexões. As diretrizes e convenções para tal são razoavelmente comuns para todas as disciplinas, o que varia é a sintaxe dos símbolos utilizados nas diferentes aplicações, tais como: elétrica, pneumática, hidráulica, etc. (McMAHON e BROWNE, 1998).

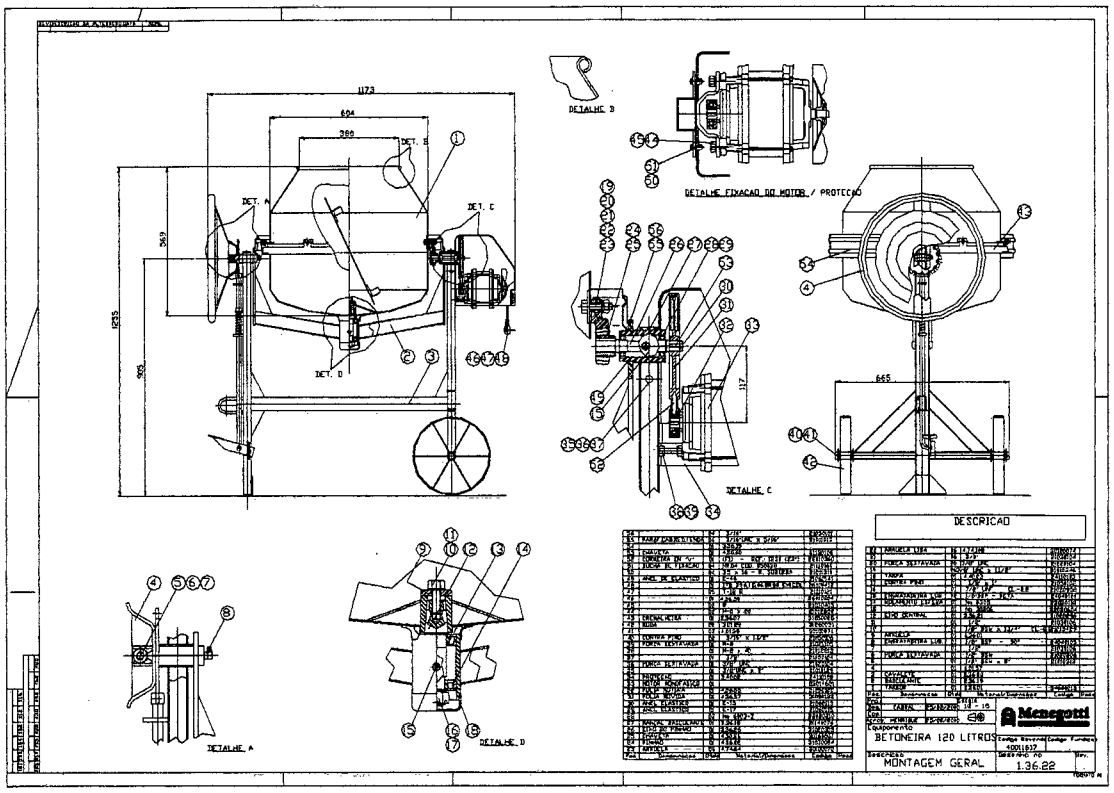


Figura 2.2: Desenho 2D - Fornecido pela Metalúrgica Erwino Menegotti Ltda.

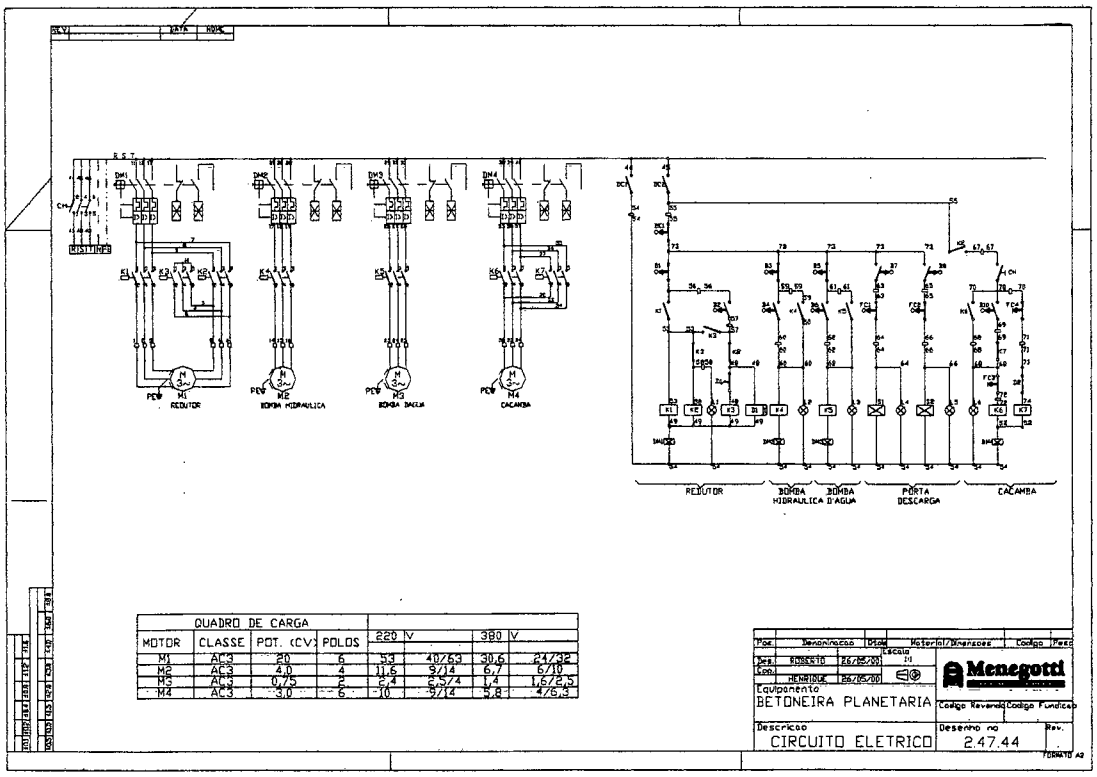


Figura 2.3: Diagrama - Fornecido pela Metalúrgica Erwino Menegotti Ltda.

Na representação de desenhos e diagramas com auxílio computacional (CAD) trabalha-se com entidades gráficas como linhas, pontos, curvas, símbolos, conexões, etc., construindo-as através de métodos variados, que fornecem ferramentas semi automáticas para edição de anotações, cotas, formas complexas, etc. e permitem a reutilização de trabalhos anteriores. Além disso, praticamente não existem limitações quanto ao tamanho do modelo. Na grande maioria dos casos, sua construção é feita em tamanho real (escala 1:1), diminuindo o número de erros e possibilitando a geração automática de cotas e outras anotações. Sinais de entrada e saída de diagramas podem ser estabelecidos, de forma que os modelos de funcionamento de um dispositivo possam ser organizados em um circuito e utilizado para simulações.

Contudo, o emprego de desenhos e diagramas convencionais possui várias restrições: sua construção exige muita habilidade, podem ser conflitantes ou ambíguos, curvas complexas são de difícil representação, sua interpretação exige o conhecimento técnico, a aquisição de informações tais como propriedades de massa ou análises de tensão são difíceis ou impossíveis, etc. As restrições inerentes à própria técnica de Projeção Ortográfica, frente à crescente necessidade de automação e integração das atividades de engenharia, estimularam o desenvolvimento de métodos de modelagem mais completos.

2.2.2 Modelos 3D

A definição de um produto envolve uma descrição detalhada de suas características. Isto significa que a representação computacional do modelo deve se aproximar ao máximo do que acontece no mundo real. Neste sentido, muito esforço foi dedicado à modelagem tridimensional. Os modelos 3D disponibilizam a representação de forma suficientemente completa e não ambígua. Eles possibilitam a representação única do desenho o que elimina os erros potenciais devido ao uso de múltiplas vistas necessárias em representações 2D. Desta forma, se tornam muito mais úteis como base de aplicações que envolvem a extração de informações de modelos para análise e manufatura (OSHUGA, 1989).

Existem três técnicas principais de modelagem 3D, as quais são usadas de forma híbrida na maioria dos sistemas CAD (McMAHON e BROWNE, 1998), (SHAH e MÄNTYLÄ, 1995) e (ZEID, 1991):

- Modelos em estrutura de arame ou *wire-frame*, Figura 2.4. Foram os primeiros a serem desenvolvidos, podendo ser entendidos como uma extensão para 3D das técnicas usadas em 2D, são a base inicial para criação de modelos mais avançados. Este método é de fácil utilização, econômico para o computador e permite a composição de símbolos, textos ou outras informações relativas a aplicações específicas. Contudo, apresentam várias deficiências que restringem sua utilização.

Eles são visualmente ambíguos, e o cálculo de propriedades mecânicas e interseções geométricas é bastante limitado.

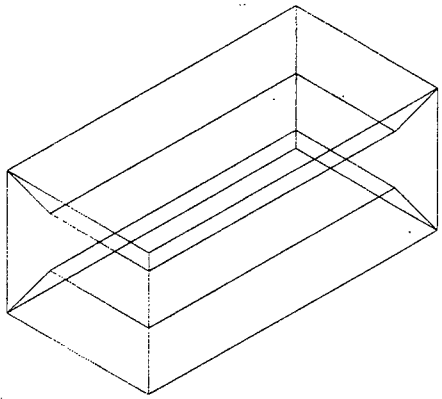


Figura 2.4 Modelo em estrutura de arame (McMAHON e BROWNE, 1998)

- Modelos de superfícies, Figura 2.5. As superfícies são modeladas em malhas de curvas paramétricas definidas em matrizes de pontos de controle e são invariantes a transformações geométricas (translação e rotação), escalonamento e distorção. Disponibilizam excelentes meios de gerar informações para manufatura e análises. Com elas, formas complexas podem ser criadas e compartilhadas entre todo pessoal presente no sistema de manufatura, sem que seja necessário grandes habilidades de interpretação. Contudo, assim como nos modelos em estruturas de arame, não existe nada que previna ambigüidades ou erros e não há identificação de partes sólidas. Além disso não existe conectividade entre superfícies diferentes de modo que alterações devem ser propagadas manualmente.

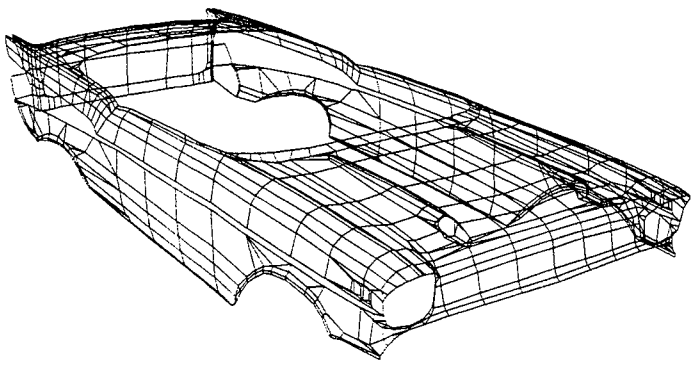
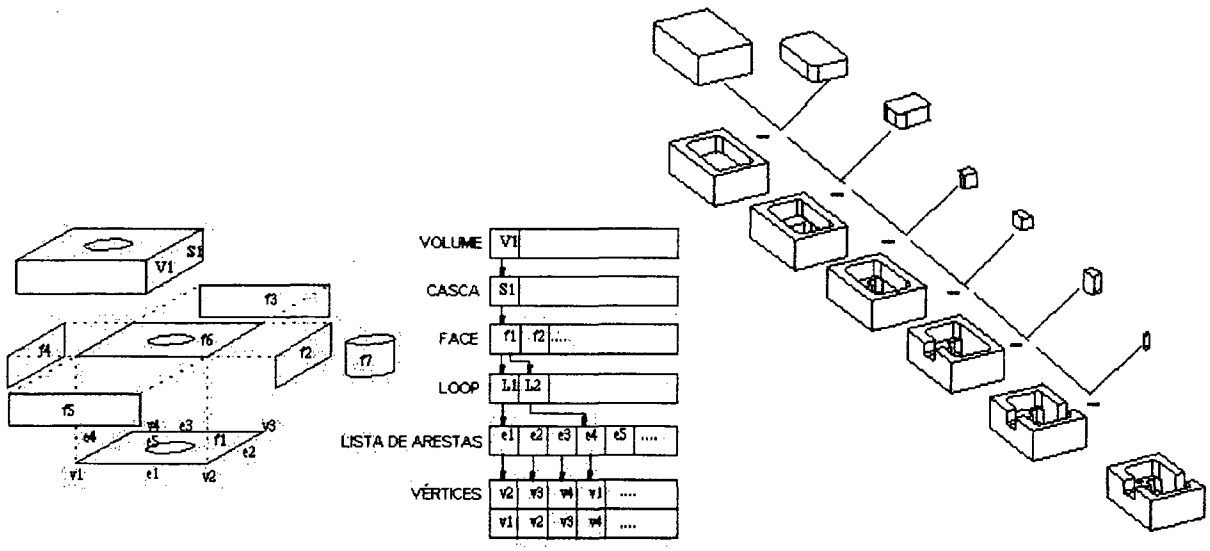


Figura 2.5: Modelo de superfície (lataria da Chevy do AutoCAD R14)

- Modelos sólidos. É a técnica de representação escolhida para a maioria das aplicações avançadas de sistemas CAD. Modelos sólidos não descrevem somente as fronteiras, como no caso do modelo de superfície, mas também as regiões internas e

externas. Dentre as técnicas de modelamento sólido, os tipos mais importantes de representação, são: Modelos B-Rep (*Boundary Representation*) Manifold Figura 2.6-a ou Não-Manifold, os quais descrevem os sólido em termos de vértices, arestas e faces; Modelos CSG (*Constructive Solid Geometric*), Figura 2.6-b, que representam o objeto em termos de combinação de formas primitivas (cubos, cilindros, cones, etc.) através de operações booleanas (união, subtração e interseção) e transformações geométricas (translação, rotação, escalonamento e distorção).



(a) Modelos B-Rep (b) Modelos CSG

Figura 2.6: Modelos sólidos (SALOMONS, 1995)

2.2.3 2D x 3D

A escolha pelo uso de uma técnica de modelamento específica, depende dos recursos disponibilizados pelo sistema CAD adotado. Com relação a este aspecto, estes sistemas podem ser classificados em três categorias:

- Sistemas CAD de grande porte: possuem os recursos mais avançados do mercado. Eles são otimizados para modelamentos 3D e geralmente oferecem ferramentas para trabalho com sólidos e superfícies de maneira unificada (modeladores híbridos). Eles são conceitualmente diferentes dos demais, pois sua estrutura de dados é idealizada e desenvolvida para servir como um elemento central de integração entre grupos de diferentes aplicações ou sistemas, tais como: CAE, CAM, KBE, etc. São exemplos: ProEngineer, CATIA e Unigraphis.
- Sistemas CAD de médio porte: Também possuem recursos avançados para modelamento 3D de sólidos e superfícies, mas nem sempre são otimizados para tal. Estes sistemas CAD também podem ser utilizados como módulos centrais para integração com outros aplicativos, contudo não são idealizados para este fim e

consequentemente, sua estrutura de dados interna é menos abrangente e detalhada que a dos sistemas CAD de grande porte. São exemplos: MicroStation Modeler, Mechanical Desktop e Solid Works.

- Sistemas CAD de pequeno porte: Estes sistemas CAD são fundamentalmente destinados à modelagem 2D. Algumas vezes possuem recursos para modelamentos de sólidos e superfícies, mas são tão escassos que acabam inviabilizando seu uso em larga escala. Além disso, sua base de dados é bastante simplificada e dificilmente podem ser usados como elemento central para integração com outros aplicativos. São exemplos: AutoCAD, ItisCAD e UniCAD.

Apesar das vantagens da representação de projeto através de modelos sólidos a grande maioria das empresas ainda utilizam desenhos 2D. Em uma pesquisa realizada pela Dataquest em 1999 (DATAQUEST, 1999), foi verificado que mais da metade das empresas dos EUA que utilizam algum tipo de sistema CAD, o fazem com técnicas 2D. A principal razão para isto é que os desenhos 2D atendem satisfatoriamente a maior parte das necessidades de documentação de projeto. Contudo esta mesma pesquisa constatou que 75% destas empresas acreditam que o modelamento 3D se tornará a principal técnica de documentação de projeto dentro de 2 anos. Não se encontrou nenhuma pesquisa neste sentido para o mercado brasileiro, entretanto através de contatos com empresas da região e revendedores de sistemas CAD, também foi observado uma clara tendência de mudança das técnicas 2D para as 3D.

2.3 ESTRUTURA DOS DADOS QUE REPRESENTAM AS INFORMAÇÕES DE PROJETO

Apesar das várias mudanças ocorridas nos sistemas CAD desde suas primeiras aplicações, sua estrutura de dados continua sendo fundamentalmente centrada em entidades geométricas. As informações de projeto são representadas em termos de entidades como: linhas, arcos e círculos, associadas à atributos de: cor, estilo, etc. e opcionalmente, dados não gráficos (*Extended Entity Data* - EED) ou entidades de associação com banco de dados externo.

Formalmente, uma estrutura de dados é definida como um grupo de entidades interligadas entre si por uma série de relacionamentos. Geralmente elas são organizadas numa ordem seqüencial, com o último elemento criado ocupando o fim da estrutura. Sob o ponto de vista de sistemas CAD, ela é um esquema lógico ou uma seqüência de passos desenvolvidos para obter determinadas representações gráficas, não-gráficas e/ou um

objetivo programado. Cada entidade requer diferentes tipos (inteiros, reais, lógicos, caracteres, etc.) e quantidades de dados (ZEID, 1991).

O nível de relacionamento ou associação entre as entidades é de particular importância, especialmente para a modelagem de sólidos. Em sistemas mais sofisticados, é possível associar dados do modelo com atributos extras (os EEDs), para aplicações especializadas tais como: a geração automática de listas de materiais, representações para análises de tensão e controle de máquinas de comando numérico [2].

2.4 ARMAZENAMENTO E RECUPERAÇÃO DE DADOS DE PROJETO EM SISTEMAS CAD

Todas as informações de projeto representadas por intermédio de sistemas CAD devem ser armazenadas em uma base de dados que possa ser facilmente recuperada pelas próprias ferramentas do sistema, ou por outras aplicações. A estrutura e o método de gerenciamento da base de dados dos sistemas CAD determinam suas principais características, no que diz respeito à qualidade, velocidade e facilidade de recuperação de informações.

A base de dados dos sistemas CAD contém grandes quantidades de tipos de dados com complexos inter-relacionamentos, devido à variedade de propriedades relacionadas ao produto que necessitam ser modeladas. STALEY, S. M e ANDERSON, D. C. (1986), fizeram uma compilação de vários esforços de pesquisadores inseridos nesta área de estudo e especificaram 20 requisitos funcionais de uma base de dados para sistemas CAD. Apesar de ser um trabalho antigo, estes requisitos não estão ultrapassados, pelo contrário, muitos deles ainda não foram completamente resolvidos. Abaixo são citados alguns destes requisitos, considerados como mais relevantes:

- Suportar múltiplos programas de aplicação de engenharia;
- Suportar a modificação e extensão dinâmica da estrutura da base de dados;
- Suportar a natureza interativa do projeto;
- Suportar a incorporação de informações semânticas na base de dados (informações que descrevem o significado dos dados armazenados);
- Não impor restrições na seqüência do projeto;
- Manter a consistência dos dados automaticamente;
- Suportar versões de projeto e múltiplos níveis de detalhe;

[2] Algumas vezes, os vendedores de sistemas CAD se referem a este tipo de associação de dados como um meio de adicionar inteligência ao modelo. Na verdade não se adiciona inteligência, mas sim, atributos específicos que possibilitam o desenvolvimento de aplicações especializadas, supostamente mais inteligentes.

- Suportar uma interface com linguagens de programação amplamente utilizadas;
- Suportar o processamento personalizado de consultas;
- Suportar o trabalho em rede de computadores;
- Suportar usuários múltiplos e simultâneos;
- Suportar múltiplas representações.

A satisfação destes requisitos funcionais é um processo evolutivo. Os primeiros sistemas CAD tinham bases de dados fundamentadas em modelos de arquivos simples sem qualquer padronização de relacionamento entre as entidades. Depois surgiram os modelos hierárquicos, modelos de rede de dados (*networks*), modelos relacionais e por último os modelos orientados a objeto. Os sistemas CAD atuais exploram cada vez mais as potencialidades dos modelos orientados a objeto (OODB). Estes modelos facilitam a representação de informações mais completas, pois são capazes de modelar objetos complexos de forma compacta e organizada.

O fácil acesso às informações da base de dados é um aspecto fundamental. Deste modo os Sistemas de Gerênciamento de Base de Dados (DBMS), são módulos fundamentais dos sistemas CAD. Os DBMS são aplicativos que permitem acesso para usar e/ou modificar os dados armazenados em uma base de dados. São responsáveis pela criação de arquivos, verificação de usuários ilegais, sincronização de acessos, proteção, verificação da unicidade, controle de versões, e outras tarefas relacionadas (ZEID, 1991). Eles formam uma camada entre a base de dados física e a interface gráfica do sistema CAD na qual atua o usuário, conforme mostra a Figura 2.7.

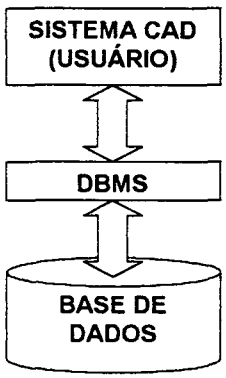


Figura 2.7: DBMS em sistemas CAD

Devido à presença da "interface DBMS", é possível extrair informações necessitadas por aplicativos integrados ao CAD, sem exigir conhecimentos extensivos sobre a estrutura de dados, fato extremamente importante na personalização do sistema CAD para projetos específicos. Para tal o sistema CAD disponibiliza linguagens de programação próprias, desenvolvidas com base em outras amplamente usadas, tais como

C, C++, Java, etc. Dentre várias funcionalidades, tais linguagens conseguem trabalhar de maneira integrada com o DBMS do sistema CAD, para acessar e manipular entidades presentes em sua base de dados. Exemplos conhecidos destas linguagens são: AutoLisp (AutoCAD), MDL (*MicroStaion Development Language*), JMDL (*Java MicroStation Development Language*), I-DEAL (*I-DEAS's Programing Language*), etc. Estas funcionalidades podem ser comparadas à conhecida linguagem de consulta a banco de dados relacionais denominada SQL.

2.5 TROCA DE DADOS DE PROJETO COM SISTEMAS CAD

A necessidade de troca de dados aparece quando informações do produto devem ser movidas entre aplicações computacionais de diferentes estações de trabalho, equipes de engenharia, departamentos ou companhias. Para melhor uso dos recursos humanos e materiais de uma empresa, o fluxo dos dados necessários a cada etapa do processo de desenvolvimento do produto deve ocorrer de forma rápida, precisa e flexível, permitindo que as tomadas de decisões sejam feitas em cooperação. Traduções de dados devem ser minimizadas a fim de preservar os dados originais tanto quanto possível, evitando eventuais perdas ou distorções de informação.

O que se busca são sistemas CAD que facilitem a troca das informações procuradas pelo projetista, de maneira automatizada e inteligente, superando as restrições causadas pela natureza complexa, diversificada e abstrata dos dados envolvidos no projeto. Com relação a estes aspectos existem três tópicos básicos relacionados: padrões para representação, troca de dados do produto e trabalho em redes de computadores e interação com outros aplicativos.

2.5.1 Padrões para troca de dados do produto

Existem duas abordagens distintas para transmissão de dados entre sistemas incompatíveis. Na primeira a tradução é direta, ou seja, por intermédio de um software tradutor os dados armazenados em um formato são mapeados para outro em um único passo. A segunda proposta é mais inteligente e geral. Ela adota a filosofia de criar uma estrutura de dados neutra, pública e independente de qualquer sistema. Neste caso, são necessários dois software tradutores, os quais são chamados de pré-processadores (traduzem o formato nativo para o neutro) e pós-processadores (traduzem o formato neutro para o nativo).

A proposta de um padrão neutro é mais atraente. No entanto, seu desenvolvimento sempre demandou grandes esforços. Isso exige a superação de barreiras

tais como: incompatibilidade entre entidades representativas existentes na estrutura de dados de diferentes sistemas, complexidade dos sistemas CAD, acesso restrito a informações sobre as bases de dados de propriedade particular e acelerado desenvolvimento da tecnologia.

A formatação dos dados destinada a comunicar informações do produto entre sistemas CAD, deve idealmente englobar definições completas sobre o produto, de forma compacta e clara. Ciente da dificuldade de realização desta tarefa, os primeiros esforços se concentraram na transferência de dados geométricos, desenvolvendo formatos neutros baseados na geometria e topologia, tais como IGES (EUA), SET (França), VDAFS (Alemanha), etc. (McMAHON e BROWNE, 1998), com este mesmo intuito, algumas empresas privadas criaram padrões de dados abertos, dentre eles o DXF (*Data Exchange Format*) (McMAHON e BROWNE, 1998). Outras iniciativas comerciais têm conseguido promover uma certa padronização na representação de dados geométricos, através do uso comum de núcleos modeladores em diferentes sistemas, tais como ACIS da Spatial Tecnology e Parasolid da Unigraphics.

Contudo, com a constante evolução da tecnologia CAD, tem-se aumentado a complexidade dos modelos produzidos. Desta forma, cresce também a necessidade por padrões para troca e armazenamento de uma maior variedade de informações. Tais padrões são chamados de formatos de dados baseados no produto. Os primeiros a surgirem foram: PDDI (EUA), PDES (EUA) e CAD*I (ESPRIT - Europa), (McMAHON e BROWNE, 1998), (SSEMAKULA e SATSANGI, 1990) e (AN et al., 1995). Apesar da importância e constantes melhoramentos destes padrões, eles ainda permanecem incompletos e sem uma aceitação ampla e irrestrita, não sendo capazes de eliminar problemas básicos como:

- Sempre existe a possibilidade de ocorrer erros numéricos em algumas conversões.
- Sempre existe a possibilidade de interpretações ambíguas das definições dos padrões genéricos por parte dos desenvolvedores de software tradutores.
- Alguns sistemas não suportam entidades do padrão genérico, criando a necessidade de filtros. Aumenta-se desta forma a complexidade dos tradutores e em contrapartida, diminui a eficiência.

O desenvolvimento do STEP (*Standard for Exchange of Product Model Data*) (McMAHON e BROWNE, 1998), (KERN, BOHN e BARCIA, 1996) e (KERN e BOHN, 1995), nome não oficial da norma ISO 10303, tem o intuito de unir os esforços dos trabalhos passados e superar estas limitações. O comitê organizador da norma ISO 10303 tem o objetivo de criar um padrão internacional de troca de dados, aberto e extensível, capaz de suportar definições do produto através de todo o seu ciclo de vida.

2.5.2 Trabalho em redes de computadores

A troca de informações de projeto também deve auxiliar o projetista a tomar suas decisões de forma rápida e segura. Tais informações devem ser capazes de responder questões sobre normas internas, práticas e procedimentos usualmente adotados na empresa, processos de manufatura, existência de projetos similares, etc. Normalmente, estas questões podem ser respondidas com dados que combinam texto, figuras e números organizados de maneira informal, presentes em guias de projeto, livros texto, catálogos de equipamentos e serviços, registros de desenhos, etc. Deste modo os sistemas CAD atuais tem sido equipado com recursos que permitem o compartilhamento de dados nos mais variados formatos, num ambiente corporativo baseado em redes de computadores.

A utilização de redes de computadores é uma questão fundamental quando se projeta um produto onde são necessárias freqüentes mudanças e reavaliações. Nestas ocasiões todos os membros da equipe de projeto são afetados e normalmente, o pessoal envolvido nem sempre se localiza em regiões próximas. Sem o auxílio de redes computacionais capazes de possibilitar o livre fluxo de informações entre localidades distantes, dentro ou fora da companhia, o desenvolvimento do projeto de um produto pode se tornar um tanto inconveniente e caro, fato ainda mais evidente nos ambientes de Engenharia Simultânea.

Na área de desenvolvimento de comunicação em rede a WWW e a Internet são as principais tecnologias envolvidas. Elas combinam um grupo de protocolos e convenções usando redes de computadores e recursos multimídia. Seus elementos construtivos principais incluem (McMAHON e BROWNE, 1998):

- Arquitetura cliente-servidor. Os documentos são processados pelo cliente através de "Web browsers" e obtidos de repositórios gerenciados por servidores da Web.
- Hipertextos. Oferece um meio atrativo para apresentação de informações através da coleção de partes discretas de textos e gráficos, dentro do qual os componentes individuais são conectados através de relacionamentos em rede (*hyperlink*). São escritos utilizando HTML (*Hypertext Markup Language*) e transferido entre cliente e servidor através do protocolo HTTP (*Hypertext Transfer Protocol*). Pode-se também transferir informações através de outros protocolos tais como FTP, Gopher, etc.
- URLs (*Uniform Resource Locators*). Usadas para localização de documentos e outras informações.

Várias companhias têm escolhido a Internet como meio de comunicação entre clientes, fornecedores ou projetistas. Desenvolvedores de software se preocupam com o estabelecimento de formatos de dados mais compactos a fim de agilizar a troca de dados de engenharia em redes de computadores. A tecnologia da Web liga todos em uma plataforma independente.

McMAHON, C. e BROWNE, J (1998), apresentaram três desenvolvimentos proporcionados pela WWW mais importantes sob o ponto de vista das aplicações CAD:

- *Plug-ins*: peças de software que permitem o trabalho com vídeos, sons, e outros formatos de arquivos em um browser tipo Netscape ou Explorer.
- Java®: linguagem de programação orientada ao objeto desenvolvida pela Sun Microsystems, que possibilita a construção de aplicativos (*Applets*) independentes de hardware que disponibilizam interatividade numa página da Web.
- VRML (*Virtual Reality Modeling Language*): permite a criação de modelos 3D que podem ser acessados via Internet e exibidos usando um visualizador tridimensional em um browser.

A implicação destas ferramentas é crucial. No passado, eram requeridos aplicativos caros para visualizar modelos CAD, desta forma limitava-se o uso de modelos tridimensionais para comunicação com clientes, fornecedores ou mesmo usuários do sistema. Com VRML e Java®, podem ser desenvolvidos *Plug-ins* de baixo custo, tornando possível visualizar, manipular e editar modelos 3D através de redes de computadores. Alguns software de CAD já permitem que o usuário exporte a informação criada em HTML, VRML, CGM e outros formatos para avaliação de equipes multifuncionais. Companhias estão escrevendo aplicativos baseados em Java®, que possibilitam acessar, gerenciar e criar remotamente, informações estabelecidas na base de dados de sistemas CAD. O desafio atual é identificar novos modos de explorar estas tecnologias.

2.5.3 Interação com outros aplicativos

O sistema CAD é apenas uma das ferramentas computacionais empregadas para o auxílio ao projeto de produtos. Aplicações como planilhas, banco de dados e editores de texto também são usadas para execução de cálculos, análises, relatórios, etc. Assim, a documentação de um projeto como um todo acaba sendo constituída de informações que são interrelacionadas porém, registradas em vários formatos de arquivos, cada qual acessado por aplicativos específicos. Por este motivo, vários sistemas CAD possuem recursos padronizados que permitem sua interação com outros aplicativos.

Estes recursos são dependentes do sistema operacional usado. Atualmente a maioria dos sistemas CAD, dirigidos para pequena e média empresa, rodam em PC com a plataforma MS Windows. Os aplicativos MS Windows, podem interagir com grande facilidade através de três protocolos padronizados: Clipboard, OLE e DDE. Com estes protocolos é possível estabelecer ligações entre diferentes documentos de projeto, enviar dados para o sistema CAD a fim de gerar desenhos de projeto a partir de parâmetros calculados por outros aplicativos, etc.

Estes protocolos são implementados de maneira diferente pelo desenvolvedor do aplicativo. A seguir tem-se uma descrição em termos gerais da funcionalidade de cada um (SAHAI, 1999):

- Clipboard: Permite transmitir dados de uma aplicação fonte para outra destino através de comandos "Cortar" e "Colar" (*Cut and Paste*). O Clipboard suporta dados em vários formatos tais como textos, imagens, áudio e vídeo. Contudo, cada aplicativo suporta alguns tipos de dados específicos.
- OLE: É uma extensão do Clipboard, na qual um *link* para a aplicação fonte é adicionada junto com os dados passados para a aplicação destino. Deste modo, quando os dados são alterados na aplicação fonte, eles são automaticamente atualizados na aplicação destino.
- DDE: Os DDE são mensagens que podem ser trocadas entre dois aplicativos que suportam este recurso, através da ação de macros geralmente escritas com a linguagem Basic.

2.6 BREVE ANÁLISE DE TÉCNICAS EMPREGADAS NO DESENVOLVIMENTO DE NOVOS SISTEMAS CAD

Os sistemas CAD atuais disponibilizam representações de forma e estrutura suficientemente completa e não ambígua. Contudo, reconhece-se que os mesmos requerem capacidades adicionais para satisfazer as necessidades de integração das novas estratégias de desenvolvimento do produto. Quatro são as razões principais para isso (SILVA Jr. e DIAS, 2000):

- Dificuldade de se integrar sistemas especialistas ou modelos de análise com modelos geométricos;
- falta de técnicas de representação padrão de projeto, planejamento e manufatura;
- dificuldade de computação de dados abstratos;
- as bases de dados não suportam todas informações necessárias nas demais atividades do ciclo de desenvolvimento do produto.

Para superar estas limitações, é necessário se desenvolver estruturas de dados mais abrangentes e métodos eficientes para utilização das mesmas. Com esse intuito, técnicas como: orientação a objeto, inteligência artificial (AI), modelamento paramétrico ou variacional, tem sido utilizadas no desenvolvimento dos novos sistemas CAD. Segue um maior detalhamento de cada uma destas técnicas.

2.6.1 Orientação a objeto

A orientação ao objeto é um estilo de programação computacional que facilita a manutenção e reutilização do software. No tradicional estilo procedural de programação, os dados são movidos ao longo de uma série de procedimentos que atuam sobre eles. Na programação orientada a objeto, cada objeto contém seus próprios dados e comunicam entre si através de troca de mensagens. O software orientado a objeto é constituído de módulos (ou objetos), que possuem comportamento próprio podendo ser facilmente retirados e modificados, ou quando desejar, pode ser introduzido novos módulos com a mesma facilidade.

Os princípios da orientação a objeto nasceram em 1960 com o desenvolvimento do Simula67, uma linguagem de simulação de eventos discretos. Mas foi em 1970, com a linguagem Smalltalk da Xerox Palo Alto Resarch Center (Xerox PARC), que seus fundamentos foram plenamente implementados. Tais princípios se baseiam em quatro conceitos fundamentais:

- Encapsulamento: É definido como uma técnica para minimizar a interdependência entre módulos, através da definição de interfaces externas. Mudanças na implementação de um módulo que preserve interface externa, não afeta a definição dos outros módulos.
- Herança: A orientação a objeto utiliza o relacionamento de herança para criar uma hierarquia de classes ou tipos de objetos. Nesta hierarquia de classes a classe filho pode herdar dados e métodos de sua ou suas classes pai, mas continuará tendo seus dados e métodos privativos. A maior vantagem da hierarquia é a oportunidade oferecida para reutilização de classes existentes em novos contextos.
- Polimorfismo: É a propriedade que um objeto possui de responder apropriadamente de acordo com sua classe, para uma mensagem padrão. Isto reduz o volume total de código, ajuda na legibilidade e possibilita a adição de novas classes ou objetos sem alterar o código anteriormente implementado.
- Abstração: Os objetos possuem comportamentos definidos por dados e métodos. Com a abstração destes elementos cada objeto passa a ter um comportamento bem definido, clarificando a leitura do problema implementado.

Atualmente a orientação a objeto é a abordagem escolhida para o desenvolvimento de sistemas CAD. Tais sistemas são constituídos por um grande volume de programas computacionais (na ordem de 100MB), o que torna mais crítica as questões de manutenção e reutilização. Com a orientação a objeto em CAD novos tipos de entidades podem ser adicionadas sem dificuldade, herdando procedimentos já desenvolvidos. A facilidade de passar mensagens entre os objetos amplia a possibilidade de associações entre diferentes entidades. Alguns trabalhos argumentam que a aplicação desta técnicas

fará com que os sistemas CAD deixem de ser usados como pranchetas eletrônicas e passem a permitir o modelamento de objetos do mundo real, facilitando também a implementação de aplicações automatizadas.

2.6.2 Inteligência artificial

Existem diversas pesquisas padrões na aplicação de inteligência artificial (AI). Apenas uma parte delas tentam desenvolver sistemas computacionais que simulam o comportamento inteligente. Esta linha de pesquisa envolve o desenvolvimento dos sistemas especialistas ou sistemas baseados em conhecimento, os quais são capazes de representar conhecimentos heurísticos de difícil expressão, através de abordagens matemáticas tradicionais.

Os sistemas baseados em conhecimento possuem uma representação formal e explícita de informações relacionadas a uma área de atividade específica, armazenada em uma base de conhecimento. O processo de aquisição de informações para compor a base de conhecimento é feito através do estudo sistemático de métodos de solução de problemas utilizados por especialistas ou através de entrevistas técnicas. A busca e extração das informações é realizada por um módulo do sistema, conhecido como máquina de inferência.

Várias são as técnicas utilizadas para representação e armazenamento do conhecimento. As conhecidas como regras de produção, estruturas (*frames*), grafos e redes de comunicação, podem ser usadas para descrever conhecimentos de engenharia. Outro grupo objetiva a captura automática de conhecimento, para tal utiliza uma espécie de rede de comunicação computacional inteligente conhecida como redes neurais. Existem ainda técnicas que visam incorporar conhecimentos incertos, são os chamados sistemas nebulosos (*fuzzy systems*).

Sistemas baseados em conhecimento têm sido aplicados para auxiliar o projetista em diferentes problemas presente em seu trabalho. Tais problemas envolvem tipicamente a seleção de materiais, processos de fabricação e técnicas analíticas, diagnósticos e avaliações. Também são empregados para auxiliar o gerenciamento de projetos complexos (método de decomposição e método de seleção e refinamento de planejamentos de trabalho), geração de soluções alternativas de projeto (método do raciocínio baseado em restrições), recuperação de informações (método do raciocínio baseado em casos), e outras áreas as quais, junto a estas aqui relatadas, são discutidas com maiores detalhes por McMAHON, C. e BROWNE, J. (1998).

A aplicação mais difundida de sistemas baseados em conhecimentos em engenharia é conhecida como Engenharia Baseada em Conhecimento (KBE – *Knowledge-Based Engineering*), uma combinação de programação orientada a objeto, Inteligência Artificial e CAD. A base do KBE é o desenvolvimento do modelo do produto utilizando

técnicas de orientação a objeto. Neste modelo, as partes do projeto são definidas como classes de objetos e estruturas em árvore do produto, e são usadas para identificar a decomposição hierárquica de montagens e sub-montagens. Qualquer tipo de informação pode ser anexada ao modelo utilizando atributos. O KBE incorpora técnicas padronizadas de modelamento geométrico à base de conhecimentos. Isto, diferentemente de sistemas CAD convencionais, não força o desenvolvimento de aplicação automatizadas com abordagens centradas no raciocínio geométrico.

2.6.3 Modelos paramétricos e variacionais

No processo de modelamento em sistemas CAD existem várias situações em que se deseja flexibilidade dimensional de entidades geométricas. Esta demanda é especialmente verdadeira quando se trabalha com famílias de produtos, peças padronizadas e nas etapas iniciais de projeto, quando não se tem conhecimento de dimensões exatas e deseja-se analisar vários arranjos de uma mesma peça. Novas abordagens de projeto paramétrico têm sido desenvolvidas a fim de facilitar a reutilização de modelos previamente construídos em novas situações.

Estas abordagens envolvem um número de diferentes técnicas e são chamadas de modelos paramétricos e variacionais, os quais se apresentam de maneira bastante similar para o usuário. Ambas destinam-se às mesmas aplicações e muitas vezes os sistemas comerciais adotam abordagens híbridas. Elas permitem a descrição da geometria de modelos não somente através de números, mas também por expressões que relacionam as dimensões com variáveis, ou outros atributos parametrizados e deste modo, pode-se conseguir um certo nível de representação das intenções de projeto. Os valores das variáveis podem ser inseridas pelo usuário ou tabelados. O modelo geométrico é definido em termos de sua forma geral e topologia, e seus valores dimensionais podem ser manipulados livremente (McMAHON e BROWNE, 1998).

O processo de criação de modelos paramétricos e variacionais são similares e podem ser divididos em quatro etapas (SHAH e MÄNTYLÄ, 1995):

- O usuário constrói o modelo geométrico com os elementos e relacionamentos desejados sem se preocupar com as dimensões.
- O usuário descreve as propriedades requeridas entre as entidades do modelo em termos de restrições geométricas, especificando as relações matemáticas.
- O sistema de modelamento aplica um procedimento para solução geral das restrições. O que resulta em um modelo com as restrições declaradas satisfeitas.
- O usuário pode criar variantes do modelo pela troca dos valores das variáveis das restrições.

O que difere um modelo paramétrico de um variacional, são os procedimentos matemáticos usados para satisfação das restrições. Existem comercialmente duas abordagens principais: método construtivo e método de resolução numérica de restrições

O método construtivo é empregado em modelos paramétricos procedurais. Recebe este nome por armazenar o método construtivo do objeto em termos de uma seqüência fixa de procedimentos. Ele resolve as restrições computando os valores em função de outros previamente determinados. Variações na geometria parametrizada são alcançadas pela modificação dos parâmetros de entrada, o que resulta no recálculo de todas as restrições seguindo a seqüência de construção armazenada.

O método de resolução numérica de restrições é empregado em modelos variacionais. Nele um sistema completo de restrições é criado e resolvido simultaneamente. Portanto, a ordem de criação das restrições não é importante, como acontece no método construtivo. Aplicando este método, a construção do modelo envolve a identificação de um número de pontos característicos e dimensões que colocam limites nas possíveis localizações de cada um.

Cada método possui vantagens e desvantagens. No método construtivo, por usar um equacionamento explícito, as restrições podem ser calculadas rapidamente. Contudo, não suportam variáveis acopladas, o que diminui bastante sua flexibilidade. Além disso, ele pode exigir um esforço computacional significativo devido a necessidade de refazer todo procedimento construtivo depois de cada mudança. No método de resolução numérica de restrições, por usar equações implícitas, pode lidar com variáveis acopladas. No entanto, o cálculo das restrições é mais lento e possui capacidade limitada de lidar com especificações incompletas do modelo, devido às condições de convergência impostas por métodos numéricos de cálculo.

2.7 CONCLUSÕES

Sistemas CAD atuais são equipados com várias técnicas de modelamento geométrico tais como modelos de arame 2D e 3D, modelos de superfície e modelos sólidos. A grande maioria das empresas que utilizam algum tipo de sistema CAD o fazem com emprego de técnicas de modelamento 2D, contudo existe uma clara tendência de migração para o modelamento 3D. Isso é devido à queda nos custos de software e hardware de última geração, mas também evidencia que a criação de modelos do produto mais completos é uma real necessidade para empresas de pequeno, médio ou grande porte.

Os métodos para armazenamento e recuperação de dados, recursos para troca ou compartilhamento de informações de projeto e desenvolvimento de aplicações

personalizadas, determinam as principais características dos sistemas CAD. Portanto, estes aspectos devem ser analisados para avaliação de sua qualidade, especialmente em empresas que necessitam de trabalho corporativo e integração de diferentes sistemas computacionais.

Os sistemas CAD experimentaram um grande avanço tecnológico desde suas primeiras aplicações. Contudo, a base de dados por eles oferecidas atualmente não fornecem todas as capacidades de representação de informações requisitadas pelas empresas. Técnicas como programação orientada a objeto, inteligência artificial e modelamento paramétrico, têm sido empregadas visando melhorar tais capacidades. Através destas técnicas são introduzidas novas abordagens para o modelamento do produto com sistemas CAD. Os usuários devem estar atentos aos conceitos associados à estas novas abordagens e os desenvolvedores devem procurar conhecê-las com maior profundidade.

3 TECNOLOGIA DE *FEATURES*, UMA ABORDAGEM DE MODELAMENTO DO PRODUTO PARA A PRÓXIMA GERAÇÃO DE SISTEMAS CAD

3.1 INTRODUÇÃO

As *features* são formas geométricas reusáveis com algum significado de engenharia, sendo necessariamente dependentes da aplicação. O objetivo primordial desta tecnologia é representar informações ou dados semânticos[3] de engenharia, numa forma computável. Ela integra técnicas de modelamento geométrico com orientação a objeto, inteligência artificial, modelamento paramétrico, etc., para modelamento do produto em sistemas CAD.

Um modelo do produto baseado em *features,* tal qual o ilustrado na Figura 3.1, é uma estrutura de dados que representa uma peça ou montagem em termos de suas *features* constituintes. Como se vê na figura, estes modelos não lidam estritamente com primitivas geométricas, mas com elementos mais significativos tais como: rebaixo, furo, ranhura, protuberância, canal, etc. Estes elementos podem ser interpretados de maneira diferente, dependendo de sua aplicação.

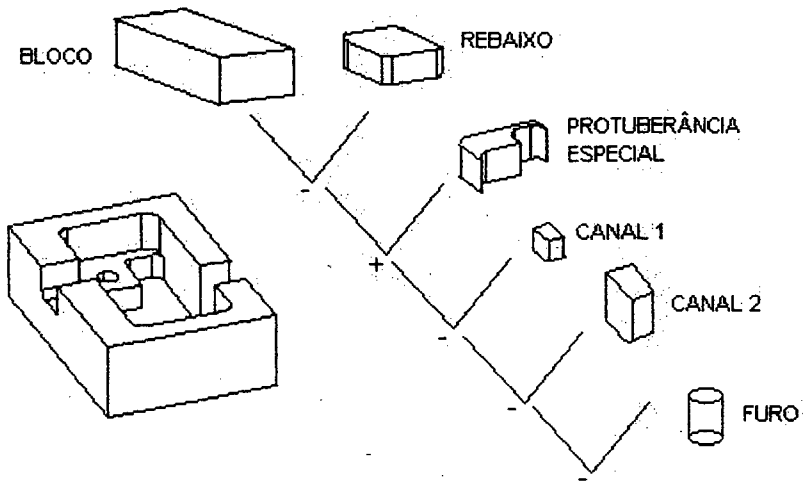


Figura 3.1: Modelo do produto baseado em *features* (SALOMONS, 1995)

[3] Informações ou dados semânticos, no caso mais geral, é a informação que descreve o significado dos dados armazenados na base de dados (STALEY e ANDERSON, 1986).

As características das *features* asseguram algumas vantagens em relação aos modelos geométricos, dentre as quais podem ser citadas (SHAH e MÄNTYLÄ, 1995) :

- Os modelos podem ser descritos usando entidades de alto nível definidos como abstrações genéricas de formas recorrentes. Relacionamentos entre as entidades caracterizam parâmetros e restrições e disponibilizam uma base para propagação de alterações no modelo.
- Conduzem à descrição do projeto em termos de entidades orientadas à aplicação, disponibilizando um meio muito melhor de representar a intenção de projeto.
- Possuem uma estrutura de múltiplos níveis de abstração. Valores previamente definidos podem ser introduzidos para suportar modelos intermediários sem especificações completas.
- Conduzem a uma interface melhorada para manipulação de geometrias de projeto. Modelos de *features* podem também realizar um trabalho de validação de alterações muito melhor, uma vez que carregam a intenção do projeto.

Devido a estas características, a tecnologia de *features* tem sido reconhecida como fundamental para implementação da Engenharia Simultânea e integração de sistemas CAD com outras ferramentas computacionais de auxílio à engenharia. "Pode-se encontrar aplicações da tecnologia de *features* em um larga variedade de atividades: projeto de peças e montagens, projeto para manufatura, geração de código CNC, planejamento de processo, planejamento de inspeção, geração de malhas de elementos finitos, etc." (SHAH e MÄNTYLÄ, 1995). Desta forma vários autores as citam como sendo o próximo passo para os modeladores geométricos.

Contudo, ainda existem aspectos que necessitam ser melhor desenvolvidos para viabilizar a plena aplicação comercial desta tecnologia. Este fato tem como causa principal a extrema dependência entre definições de *features* e sua aplicação e o fato delas representarem apenas informações preestabelecidas. Todavia, "estima-se que cerca de 80% de toda a tarefa de projeto pode ser realizada com base em trabalhos anteriores" (SHAH e MÄNTYLÄ, 1995). Por este motivo a aplicação da tecnologia de *features* tem se demonstrado viável, e sistemas CAD comerciais já apresentam algumas ferramentas desenvolvidas com base neste conceito.

Neste capítulo, a tecnologia de *features* é analisada com o objetivo de esclarecer conceitos inerentes, demonstrar suas potencialidades e limitações para desenvolvimento de aplicações computacionais de auxílio ao projeto. Inicia-se com uma revisão de aspectos gerais da tecnologia. Segue-se com uma descrição resumida dos diferentes empregos de *features* em atividades de projeto, onde também são revistas as técnicas para identificação e formalização de *features* de projeto. Por fim, faz-se uma análise dos modeladores de *features* acadêmicos e comerciais.

3.2 GENERALIDADES DA TECNOLOGIA DE *FEATURES*

Devido à grande variedade de aplicações de sistemas baseados em *features*, nem sempre os aspectos conceituais inerentes à tecnologia ficam claros. Nos próximos itens apresenta-se uma discussão que visa esclarecer terminologias e possibilitar uma visão geral da tecnologia de *features*.

3.2.1 Definições de *features*

Os trabalhos iniciais com *features* (década de 70) visavam a criação de técnicas capazes de extrair dados de modelos geométricos para geração automática do planejamento de processo, programas NC e códigos GT (SHAH, 1991). Tinha-se apenas uma visão sob o ponto de vista da fabricação. Elas representavam formas e atributos tecnológicos associados com operações e ferramentas de fabricação (BUTTERFIELD et al., 1985). Este conceito se ampliou a partir do momento em que as *features* começaram a ser empregadas em atividades de projeto. Uma alteração fundamental foi a inclusão do conceito de macro-*features, features* básicas ou iniciais (LUBY, DIXON e SIMMONS, 1986), (ROY e PANAYL, 1997), (SHAH, 1988). Estas *features* são formas independentes usadas no início do processo de modelamento tais como blocos e cilindros, ou peças completas que possam ser usadas em projetos variantes .

Portanto, a palavra *feature* na sua forma mais ampla, já não se refere apenas a detalhes das peças, tal qual o proposto nos primeiros desenvolvimentos desta tecnologia. Atualmente existem várias definições de *features* na literatura, elas variam de acordo com a intenção de aplicação do autor. Contudo, algumas são bastante genéricas e se adeqüam ao escopo do presente trabalho, dentre as quais pode-se citar:

- São estruturas de estereótipos do conhecimento, com base em processos cognitivos de projeto, análise, planejamento e qualquer outra atividade de engenharia, sendo necessariamente dependente do ponto de vista e da aplicação (SHAH e MÄNTYLÄ, 1995).
- Qualquer geometria percebida, elemento funcional ou propriedade de um objeto, útil no entendimento da função, comportamento ou performance do mesmo (McMAHON E BROWNE, 1998).

Numa tentativa de estabelecer uma maior padronização, SHAH, J. J. e MÄNTYLÄ, M. (1995) estabeleceram um paradigma. Para eles as definições devem respeitar quatro afirmações básicas:

- uma *feature* é uma constituinte física de uma parte;
- está relacionada com uma forma genérica;
- tem um significado de engenharia;

- tem propriedades previsíveis.

3.2.2 Propriedades e tipos de *features*

As *features* enriquecem a representação do produto, pois podem lidar com dados significativos tais como: aspectos funcionais, parametrização, tecnologias, etc. (MARTINO, FALCIDIENO e HABINGER, 1998). Isso é feito através do encapsulamento de uma infinidade de atributos ou propriedades. SHAH J. J. e MÄNTYLLÄ, M. (1995), descrevem uma lista exaustiva deles, dentre os quais pode-se citar:

- forma geométrica;
- parâmetros dimensionais;
- algoritmos de validação e reconhecimento;
- histórico de construção do modelo;
- modelos para FEA;
- algoritmos de mapeamento entre diferentes representações;
- atributos não geométricos, etc.

Deste modo, dependendo das propriedades encapsuladas, surgem diferentes tipos de *features*, para as quais SHAH, J. J. e MÄNTYLÄ, M. (1995) introduziram a seguinte classificação:

- *Features* geométricas:
  - *Features* de forma: Porções da geometria nominal.
  - *Features* de tolerâncias: Desvios de forma/tamanho/localização nominal.
  - *Features* de montagem: Grupos de vários tipos para definir relações de montagem.
- *Features* funcionais: Grupos relacionados com funções específicas.
- *Features* de material: Composição do material, tratamento, condição, etc.

Existe ainda a proposta de subdivisão das *features* geométricas de acordo com o objetivo da aplicação em: *features* de projeto, de manufatura, de inspeção, etc. Evidentemente as *features* de forma são as de maior interesse nos sistemas CAD, sendo este o enfoque dado neste trabalho.

3.2.3 Técnicas para criação de modelos baseados em *features*

Os modelos do produto baseados em *features* podem ser criados em modeladores desenvolvidos com base em três abordagens principais. A primeira é denominada criação interativa, nela o modelo é construído por técnicas convencionais de modelamento geométrico, então o usuário seleciona interativamente as entidades de definição da *feature*. A segunda chama-se reconhecimento de *features*, o modelo também é construído com técnicas convencionais, mas as *features* são capturadas automaticamente,

através de algoritmos fornecidos pelo sistema. A terceira é o projeto por *features*, na qual o modelo é construído a partir da instanciação de definições preestabelecidas em bibliotecas de *features* organizadas em diferentes níveis de abstração.

As técnicas de definição interativa de *features* são de mais fácil implementação e podem fornecer maior flexibilidade ao usuário, o que é bastante vantajoso quanto à personalização do ambiente de projeto, mas o processo de modelamento pode ser muito trabalhoso. Os algoritmos de reconhecimento apresentam a vantagem de possibilitarem o desenvolvimento de representações bastante específicas, todavia as implementações são geralmente muito complexas. Estas duas abordagens ainda apresentam a desvantagem de sempre criarem *features* com poucas informações. Em aplicações práticas isso força a adição manual de alguns dados. No projeto por *features* os modelos construídos podem ter um grande volume de informação, isso vai depender de como foram implementadas as definições presentes na biblioteca de *features* utilizada. Esta abordagem é criticada por limitar o processo de criação do projetista, mas este mesmo aspecto tem sido visto positivamente, uma vez que favorece a padronização e considerações sobre a manufaturabilidade, montabilidade, etc., durante o projeto.

Contudo, a tecnologia de *features* tem uma característica fundamental que conduz a sérias limitações, ela é extremamente dependente da aplicação. Deste modo, toma-se difícil o desenvolvimento de padrões abrangentes para representação e troca de dados de modelos baseados em *features* e os sistemas de modelamento acabam sendo muito específicos. Esta característica vai de encontro aos objetivos de integração da tecnologia. Na prática, busca-se minimizar as limitações através de abordagens que utilizam as várias técnicas de criação de modelos de forma híbrida.

Assim não existe consenso com relação aos aspectos organizacionais de um sistema modelador baseado em *features*. Alguns estudos propõem a utilização de um modelo central criado com técnicas de projeto por *features*, a partir do qual outros serão derivados através de técnicas de mapeamento ou conversão (BRONSVOORT e JANSEN, 1993), (DUAN et al., 1993). Outros estudos, acreditam que uma abordagem híbrida com grupos definidos de *features*, técnicas de mapeamento, reconhecimento e definição interativa trabalhando em conjunto é o cenário mais provável (MARTINO, FALCIDIENO e HAMBINGER, 1998), (McMAHON e BROWNE, 1998).

3.2.4 Validação dos modelos

Quando uma *feature* é criada, editada ou manipulada, é necessário determinar se a operação e o resultado são válidos, ou seja, se o significado ou semântica estabelecida na implementação da *feature* estão sendo respeitados. Isto não deve ser confundido com validade geométrica ou topológica do modelo, as quais são baseadas em leis matemáticas.

Features são inválidas se alguma condição declarada em sua definição for violada. Tais condições podem ser baseadas em limites de tamanho, forma, localização, orientação, dados sobre a intenção de projeto, função e significado da geometria, etc., sendo portanto, dependente do contexto e aplicação (MAZIERO, 1998).

Esta é uma questão fundamental para o desenvolvimento de um sistema genuíno de modelamento baseado em *features*. Isso porque uma das idéias básicas desta tecnologia é que informações funcionais podem ser associadas às geométricas. Contudo, esta associação irá se tornar inútil se a forma de uma *feature* que é adicionada ao modelo com uma intenção de projeto específica for livremente modificada devido a operações de modelagem subseqüente. "A modificação arbitrária da semântica de uma *feature* é desaconselhada se o desejo é que o modelamento de *features* seja realmente mais poderoso que o modelamento geométrico" (BIDARRA e BRONSVOORT, 2000).

Não existem métodos matemáticos elegantes para validação de *features*. Na implementação são empregadas técnicas de inteligência artificial de modo intensivo (MAZIERO, 1998), e muitas vezes é necessário a interação manual do usuário. Também é possível adotar critérios de validação bastante específicos e utilizá-los para avaliação de manufaturabilidade, montabilidade, custo, tolerâncias, etc.

3.2.5 Mapeamento de *features*

Uma mesma peça pode ser constituída com grupos de *features* de diferentes tipos, dependendo da tarefa de engenharia a ser executada. Para melhor se adequar ao processo de projeto, as *features* devem ser entendidas em termos de formas geométricas que satisfazem certos requisitos funcionais, para o processo de planejamento, elas devem conter informações relevantes necessárias para sua fabricação, etc. Para realizar de forma plena o propósito de integração de modelos do produto, em algumas aplicações desta tecnologia o modelador de *features* deve ser capaz realizar a conversão de definições entre estas diferentes visões. Esse processo é chamado de mapeamento, conversão, transformação ou transmutação de *features* (BRONSVOORT e JANSEN, 1993), (SHAH e MANTYLÄ).

A conversão de *features* possui poucos trabalhos relacionados. Na maioria deles, as *features* são derivadas através de módulos de conversão a partir de modelos construídos sob o enfoque do projeto (vista primária), para outros modelos secundários. Esta arquitetura faz com que modificações no modelo de projeto sejam automaticamente refletidas para as outras vistas. Numa situação ideal deseja-se sistemas que suportem a conversão bidirecional. Este tópico tem representado um grande desafio para implementação computacional, devido principalmente à dificuldade de ser estabelecer padrões de representação.

3.2.6 Representação computacional de *features*

"A representação de uma *feature* determina como suas propriedades serão organizadas para implementação computacional, ela diz respeito às *features* individualmente e ao relacionamento destas com as demais presentes no modelo" (SALOMONS, 1995). Discussões sobre métodos de representação são de grande interesse. As técnicas utilizadas ainda variam muito entre os sistemas CAD, dificultando a troca de dados. Tem-se buscado a padronização sob um consenso internacional, com o intuito de evitar este problema. Atualmente o desenvolvimento do padrão STEP (ISO 10303), representa o esforço mais promissor neste sentido. Contudo o resultado prático ainda está um tanto incerto. Deste modo, muitos pesquisadores argumentam que a padronização não deve restringir a pesquisa nesta área (SALOMONS, 1995), (SHAH e ROGERS, 1988), (SHAH e MATTHEW, 1991).

Quando se considera a representação das *features* de forma, pode-se dizer que existem duas partes básicas:

- Geometria: pode ser representada explicitamente ou implicitamente. Na representação explícita ou enumerativa, tem-se uma lista de entidades geométricas constituintes, normalmente na forma de estruturas B-rep (manifold ou não-manifold) ou CSG. Esta abordagem é mais difundida em sistemas modeladores de *features* com criação interativa ou com reconhecimento automático. Na representação implícita modela-se a informação parametricamente de modo que os dados geométricos devem ser calculados quando requisitados. Normalmente, isso é feito através de operações booleanas (árvore CSG) ou perfis paramétricos em conjunto com operações de varredura, sendo empregado em sistemas modeladores de *features* através de técnicas de projeto por *features*. (BRONSVOORT e JANSEN, 1993), (SHAH e MÄNTYLÄ, 1995).
- Dados que determinam a semântica da *feature*: para representação vêm sendo utilizadas técnicas de inteligência artificial (AI) tais como: regras *if ... else,* representações lógicas combinadas com técnicas de inferência, sistemas modulares (*Frame Systems*) e redes neurais. Outras abordagens sugerem a captura de significado de engenharia através do relacionamento geométrico entre as *features* ou através do histórico de construção do modelo.

Estas duas partes devem estar intimamente relacionadas e as relações entre as *features* constituintes do modelo também devem ser representadas, de modo que a partir do modelo se possa extrair informações suficientes para as atividades destinadas. Para tal os sistemas têm empregado abordagens que combinam pontos positivos de várias estruturas. Estruturas de grafos têm sido a abordagem de preferência na maioria dos sistemas modeladores para representação de relações internas e externas de *features*. Os modelos

paramétricos são reconhecidos como a base para o modelamento de *features*, pois os parâmetros dimensionais são de fundamental importância para o projeto e manufatura.

Alguns aspectos relacionados à representação de *features* podem ser observados no exemplo de estrutura de representação ilustrado na Figura 3.2. A figura mostra uma arquitetura de quatro níveis para representação de *features* de projeto proposta por SHAH, J. J. e ROGERS, M. (1988). O primeiro nível, "Grafo da *feature*", contém uma estrutura de grafo com dois tipos de *feature* (pai-filho), e dois tipos de relacionamento (adjacência e hierarquia). O segundo nível é a "Lista de propriedades da *feature*", com dados genéricos que funcionam como uma classe orientada ao objeto contendo parâmetros independentes e derivados e diferentes tipos de regras procedurais. O nível três, "Dados de instância", contém uma lista com os dados instanciados pelo usuário na criação da *feature*, o último nível é a "Árvore CSG" com a forma geométrica da *feature*.

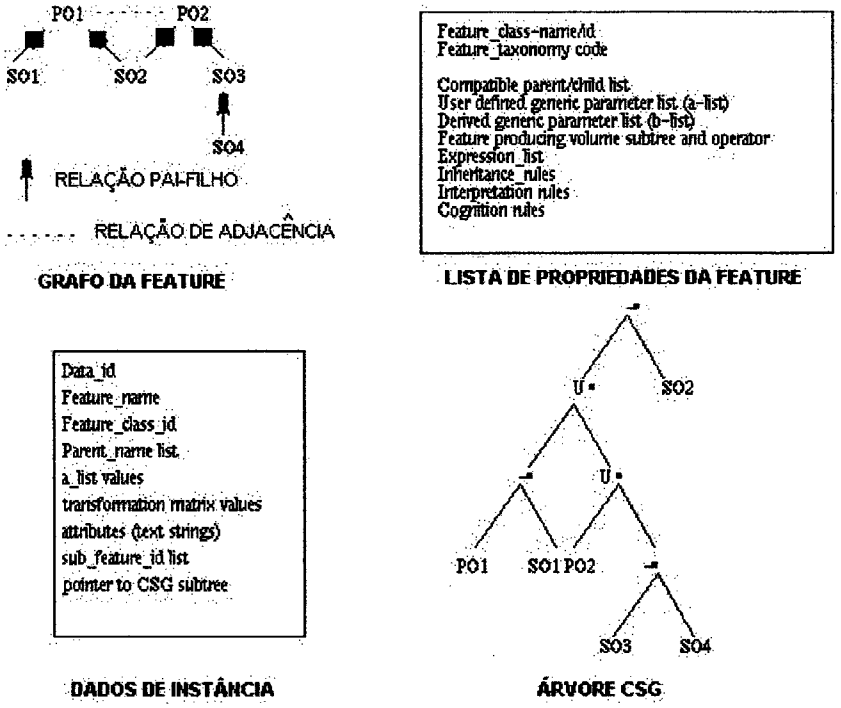


Figura 3.2: Exemplo de representação de *feature* (SHAH e ROGERS, 1988)[4]

3.2.7 Classificação das *features*

Existe um consenso em dizer que os grupos de *features* devem ser estruturados em classes pertencentes a diferentes níveis de especialização. A forma de classificação das representações é dependente da metodologia de representação, do campo e estratégia de aplicação e deve favorecer a reusabilidade e rápido acesso às informações. As classes são

[4] O texto não foi traduzido já que a idéia da figura é ilustrar uma lista de variáveis. Sem a preocupação de especificar o que são exatamente.

organizadas adequadamente quando se utilizam estruturas hierárquicas, tomando por base as técnicas de programação orientada ao objeto. Esta abordagem é desejável devido aos seguintes aspectos (McMAHON e BROWNE, 1998), (SALOMONS, 1995), (SHAH e MÄNTYLÄ, 1995):

- mecanismos generalizados podem ser projetados para suportar cada família no lugar de métodos especializados para cada ciasse separadamente;
- famílias de *feature* podem conduzir ao uso de terminologias comuns, facilitando o desenvolvimento de interfaces universais para sistemas modulares e extensão do conhecimento com *features* definidas pelo usuário;
- famílias genéricas podem ser úteis no desenvolvimento de padrões para troca de dados do produto;
- peças distintas podem ser modeladas com um mesmo grupo de representações.

Existem várias propostas para classificação de *features*. Elas podem ser baseadas em formas, aplicações ou produtos. Podem ainda, estar ligadas a intenções distintas, tal como o simples catálogo ou construção de sistemas computacionais.

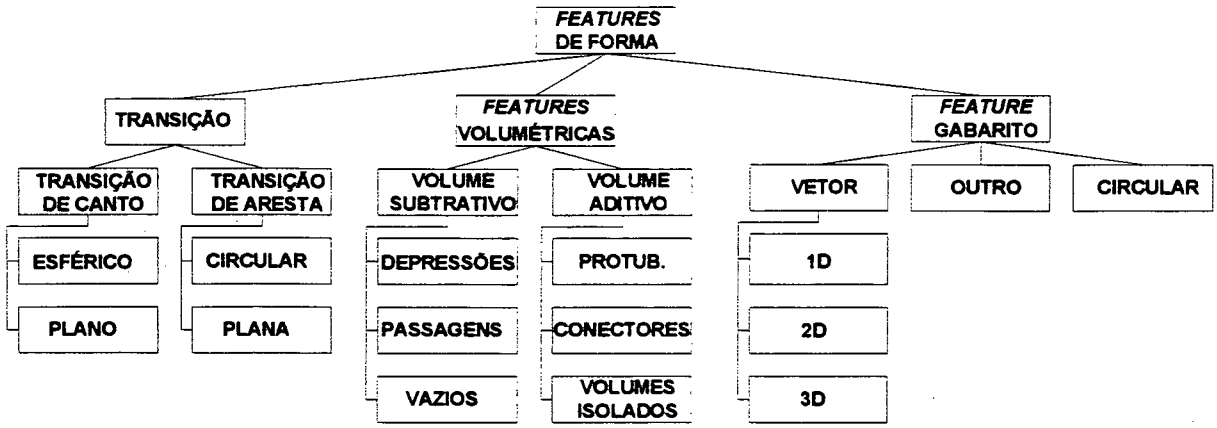


Figura 3.3: Classificação de *features* de forma do STEP (SHAH e MÄNTYLÄ, 1995)

A Figura 3.3 ilustra a classificação de *features* de forma no padrão STEP - parte 48. Deve-se ressaltar que este padrão trata as *features* como sendo uma porção da superfície de uma forma geométrica, que está em conformidade com algum modelo estereotipado e é por algum motivo, considerada uma unidade. O STEP assume que, no processo de projeto, o uso de *features* é opcional e nem toda porção do modelo necessita ser representada com uma *feature*. Esta visão está mais de acordo com a que se tinha nas primeiras aplicações da tecnologia de *features* (SHAH e MÄNTYLÄ, 1995).

Na Figura 3.3 vê-se que as *features* são divididas em três classes:

- *Features* de Volume: é um incremento ou decremento no volume da forma. Ex.: furo ou um ressalto.

- Transição: separam ou misturam superfícies. Ex.: chanfro reto ou circular.
- *Features* Gabarito: é um conjunto de *features* similares em algum arranjo geométrico regular.

A Figura 3.4 é outro exemplo de classificação, com escopo mais restrito que a do padrão STEP. Ela foi proposta por LUBY, C. DIXON, J. R. e SIMMONS, M. K. (1986) para o desenvolvimento de um sistema modelador de *features*. Considera-se que qualquer geometria do modelo deve ser necessariamente construída com base em *features*, introduzindo o conceito de macro-*features*, cuja definição já foi comentada anteriormente, no item 3.2.1.

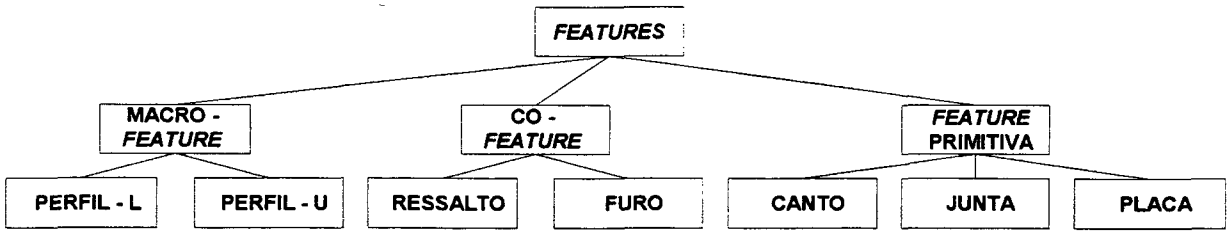


Figura 3.4: Estrutura de classificação das *features* do modelador CASPER
Adaptado de (LUBY, DIXON e SIMMONS, 1986)

Como pode ser observado na Figura 3.4, existem 3 classes de *features*:

- Macro-*features*: são classes de formas geométricas, tais como: Perfis U, Perfis L. Nesta classe pode-se adicionar outras definições tais como: blocos, tarugos, etc.
- Co-*features*: são detalhes que podem ser adicionados às macro-*features* tais como: furos, ressaltos. Também se pode adicionar outras definições nesta classe tais como: chanfros, nervuras, entalhes, rebaixos, etc.
- *Features* primitivas: são entidades de posicionamento usadas para possibilitar que uma macro-*feature* seja anexada a outra *feature* qualquer.

3.3 *FEATURES* NO PROJETO

As *features* são úteis em muitas atividades de projeto. Tradicionalmente elas têm sido empregadas na etapa de detalhamento. Nesta fase, além de melhorar o ambiente de projeto de um sistema CAD, tal qual demonstrado nas referências (DURAN et al., 1993), (FERREIRA, BUTZKE e NETO, 1997), (LIMA, 1994), (ROY e PANAYL, 1997), (SCHÜTZER, GLOCKNER e BATOCCHIO, 1999), a tecnologia de *features* também torna possível a realização de análises simultâneas utilizando sistemas numéricos ou baseados em conhecimentos. Exemplos de tais análises são: análise de tensão utilizando elementos

finitos (HETEN, 2000), avaliação de manufaturabilidade usando sistemas baseados em regras (LUBY, DIXON e SIMMONS, 1986), análise de tolerâncias (FENG et al., 1996) e estimativa de custos (LEIBL, HUNDAL e HOEHNE, 1999).

As etapas preliminares de projeto não são bem suportadas pelos atuais métodos de modelamento do produto, incluindo modelos baseados em *features*. Isso porque nesta fase os aspectos mais importantes do projeto não estão relacionados com a forma do produto e sim, com conceitos de função, comportamento e estrutura do mesmo. Para tal têm sido propostas ferramentas computacionais capazes de trabalhar com informações mais abstratas, tais como: sistemas especialistas, sistemas de projeto *top-down* e sistemas de projeto funcional. Estas propostas demonstram que a tecnologia de *features* pode exercer um importante papel.

Nos sistemas especialistas, as *features* contribuem para melhorar a abstração de entidades e fenômenos usados no projeto de engenharia, facilitando o trabalho com grupos de entidades geométricas. Nos sistemas de projeto *top-down*, as *features* podem representar geometrias abstratas relacionadas com partes do projeto, sem que haja comprometimento com soluções detalhadas. Nos sistemas para projeto funcional, cada definição de *feature* pode ser associada a uma função ou grupos de funções, criando-se um meio para mapeamento da descrição funcional do produto para sua forma (SHAH e MÄNTYLÄ, 1995).

Contudo, não existe um grupo universal de *features*, em cada aplicação é necessário identificar, formalizar e arquivar as *features* requisitadas para executar as tarefas relacionadas. As *features* direcionadas a atividades específicas devem ser desenvolvidas pela própria empresa, escolhendo definições particulares que possam ser reutilizadas com freqüência. É através da expansão dos grupos, pela construção de novas definições reusáveis em aplicações personalizadas, que o uso desta tecnologia se torna lucrativo. Portanto justifica-se um melhor entendimento do processo de identificação e formalização de *features* para o projeto (SHAH e MÄNTYLÄ, 1995).

3.3.1 Processo de identificação e formalização de *features* de projeto

Para identificação e formalização das *features* deve-se analisar as peças quanto ao produto em si e ao processo de projeto. SHAH, J. J. e MÄNTYLLÄ, M. (1995), propõem uma sistemática para tal. Nela separa-se a análise orientada ao produto da orientada ao processo de projeto, os resultados são a posterior, reconciliados e refinados. Esta sistemática descreve de forma clara um processo genérico e natural de trabalho. Ela pode ser resumidamente descrita da seguinte forma:

1) Iniciar com a análise voltada ao produto. Nela deve ser descrito os parâmetros dimensionais de formas macro simples ou compostas que:

- Sirvam a uma função de projeto ou compõem uma região funcional;
- Sejam reusáveis;
- Sejam parametrizáveis;
- Pertençam ao vocabulário do projetista.

2) Na análise voltada ao processo de projeto, deve-se:

- Identificar a *feature* base ou de referência a partir da qual o projetista inicia o modelamento da peça;
- Determinar os sub-processos de construção relacionando-os com as *features* identificadas na análise anterior;
- Determinar os parâmetros de entrada mais adequados;
- Determinar os parâmetros dependentes (derivados e normalizados), independentes e fixos (não variam nas diferentes versões das peças estudadas);
- Determinar o método de posicionamento e orientação.

3) As informações obtidas na análise voltada ao produto e ao processo de projeto devem ser agrupadas, fazendo um esboço da definição de cada *feature* com a descrição das propriedades inerentes de interesse, tais como: parâmetros dimensionais, restrições geométricas intrínsecas e extrínsecas, classificação, etc. As definições menos relevantes devem ser excluídas do grupo, para tal, os principais aspectos a serem considerandos são: freqüência de ocorrência e facilidade de criar a *feature* por técnicas modelamento padrão. A validade do trabalho como um todo deve ser analisada verificando principalmente: se o grupo de *features* atende as necessidades de modelamento para a aplicação intencionada, se as propriedades incluídas são as estritamente necessárias e se não existem definições muito parecidas.

4) As definições já identificadas e formalizadas podem ser usadas numa simulação de modelamento usando caneta e papel para obter opiniões bem fundamentadas dos seus futuros usuários. Por fim, faz se a determinação do método de representação das *features* para sua implementação computacional.

3.4 SISTEMAS MODELADORES DE *FEATURES*

Atualmente existem sistemas modeladores de *features* acadêmicos capazes de demonstrar a potencialidade da plena utilização desta tecnologia e outros sistemas modeladores de sólidos comerciais que empregam a tecnologia de *features* de modo parcial.

3.4.1 Sistemas acadêmicos

As características dos sistemas modeladores de *features* acadêmicos variam de acordo com o método de representação das definições, nível de suporte às definições do usuário, tipo de ligação com o modelador geométrico, contexto de aplicação e suporte à validação de *features*. Tais características são interdependentes, mas as principais funcionalidades do sistema são determinadas pelas três primeiras as quais são discutidas com maiores detalhes por SHAH, J. J. e MÄNTYLLÄ, M. (1995). Contudo, existem alguns requisitos básicos, que estes sistemas modeladores de features devem satisfazer. BRONSVOORT, W. F. e JANSEN, F. W. (1993) relacionam algumas delas:

- O sistema deve ser interativo e gráfico, pois esta é a melhor maneira para suportar o processo de modelamento.
- Deve existir um mecanismo para definir descrições personalizadas de *features* e armazená-las em uma biblioteca de *features*. A parametrização geométrica e topológica deve ser suportada.
- Deve existir um mecanismo para criar instâncias de uma *feature* pela especificação dos parâmetros requeridos.
- Entidades de restrições devem estar disponíveis para garantir a validade das *features* e para definir relacionamento geométrico entre diferentes *features*.
- Deve suportar o projeto *top-down* e *bottom-up*.

Atualmente, existem muitos sistemas modeladores de *features* acadêmicos. SALOMONS, O. W., VAN HOUNTEN, F. J. A. M. e KALS, H. J. J. (1993) e LIMA, C. P. (1994), fazem uma avaliação sucinta de vários deles. A seguir, como exemplo, descreve-se alguns detalhes sobre os modeladores ASU Features Testbed e SPIFF.

O ASU Feature Tesbed, com uma interface mostrada na Figura 3.5, foi um dos primeiros sistemas modeladores de *features* com propósitos mais abrangentes à ser implementado e está sendo continuamente desenvolvido pela Universidade do Estado do Arizona (*Arizona State University - ASU*) - EUA, (ASU Design Automation Lab, 2000). Ele é um conjunto de módulos de projeto, documentação e avaliação de peças mecânicas que oferecem recursos para descrição unificada de produtos. Esses módulos estão divididos em dois sistemas: um para modelamento do produto e outro para mapeamento dos modelos. O sistema de modelamento do produto é composto de vários modeladores, cada um possuindo representações particulares de *features*, direcionadas à aplicações específicas. O sistema de mapeamento permite a transição de informações entre as diferentes representações e assim, integra as informações do produto modelado (LIMA, 1994), (SHAH, 1991), (SHAH e ROGERS, 1988), (SHAH e MÄNTYLÄ, 1995), (SHAH e MATHEW, 1991).

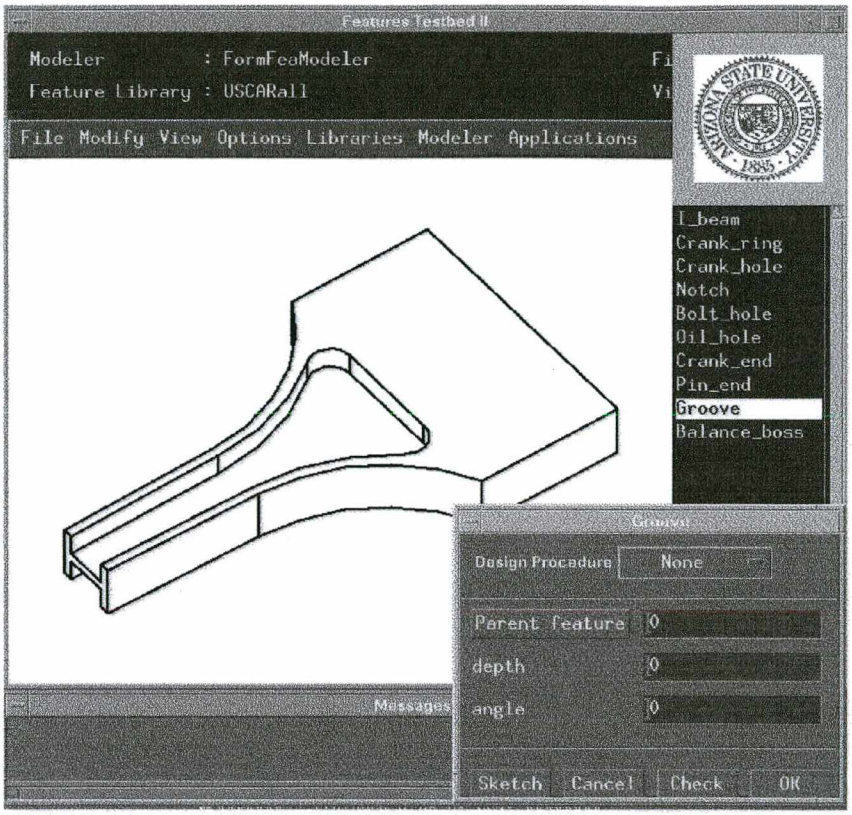


Figura 3.5: Interface do ASU Feature Testbed (ASU Design Automation Lab, 2000)

O SPIFF (*System for Product modelling by Interaction on Form Features*), com uma interface mostrada na Figura 3.6, é um desenvolvimento mais recente realizado pela Universidade de Tecnologia de Delft (*Delft University of Technology*), Holanda (ITS TU-Delft, 2000). Ele é um protótipo de sistema modelador de *features* constituído com o objetivo de experimentar novas técnicas de gerenciamento de restrições, mapeamento de definições e manutenção da validade dos modelos. O SPIFF é dividido em dois sistemas funcionais principais: O gerenciador da biblioteca de *features*, que disponibiliza ferramentas interativas para especificação das classes de *features* e para sua organização em bibliotecas de aplicações específicas e o Modelador de *features*, que disponibiliza ferramentas de modelamento para criação e manipulação de modelos de *features* com base nas definições da biblioteca (BIDARRA e BRONSVOORT, 2000).

De um modo geral, nestes sistemas o modelo é criado através da seleção e instanciação de *features* de projeto pré-definidas numa biblioteca. Os valores de parâmetros característicos tais como as dimensões principais da peça, são determinados diretamente na interface com o usuário, através do uso de caixas de diálogo, por derivação de outros parâmetros ou são previamente definidos na própria representação computacional da *feature*. O relacionamento ("pai-filho") entre as *features* que constituem o modelo é estabelecido com a determinação de entidades de referência através da identificação gráfica de faces, arestas e vértices.

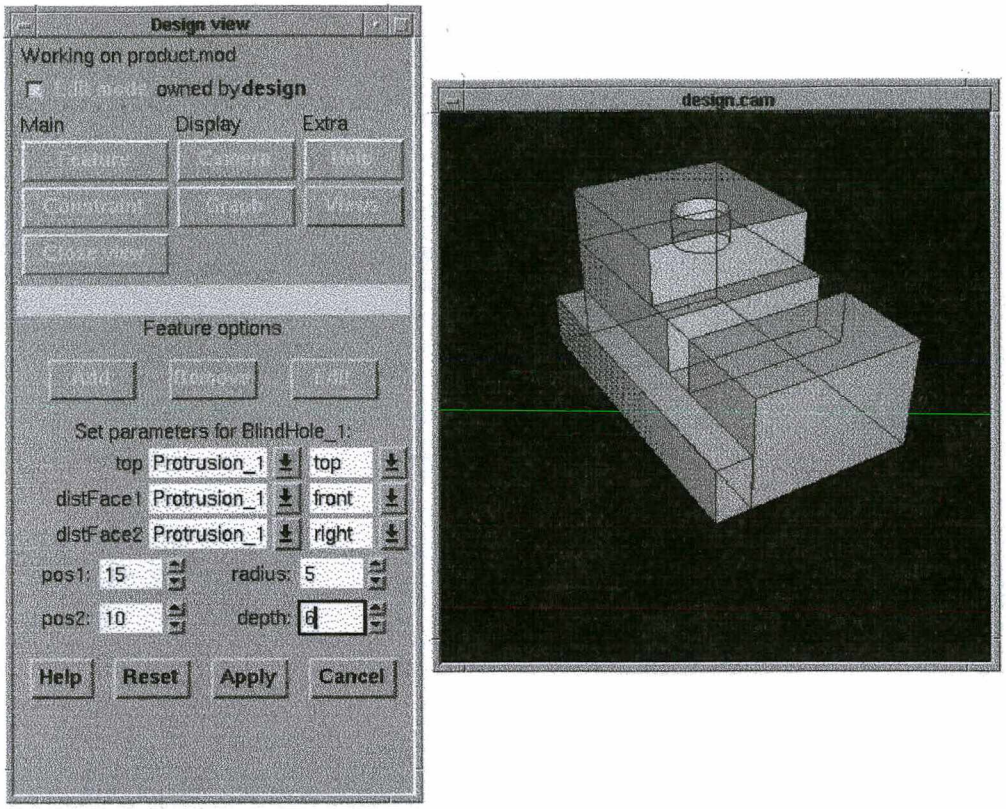


Figura 3.6: Interface do SPIFF (ITS TU-Delft, 2000)

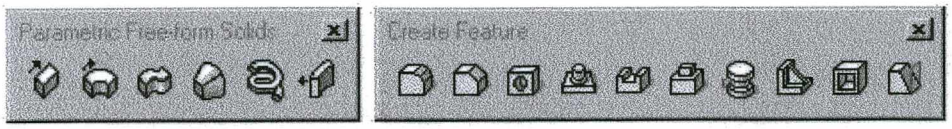
Na representação das *features,* são definidos métodos e restrições para sua definição. Elas automatizam muitas tarefas comumente realizadas pelo usuário durante a modelagem de uma peça. O relacionamento entre as entidades, permite que alterações locais se propaguem automaticamente ao longo da estrutura hierárquica de representação do modelo. Permite também que vistas explodidas de um modelo de montagem sejam geradas com um mínimo de intervenção manual. Procedimentos de manipulação podem ser automatizados, tal como a definição de coordenadas de trabalho e vista mais adequada para o posicionamento de uma determinada *feature*.

3.4.2 Sistemas comerciais

Sistemas CAD comerciais de médio ou de grande porte oferecem uma abordagem comum para modelamento geométrico de sólidos através de um grupo de ferramentas baseadas na tecnologia de *features*. A Figura 3.7 mostra exemplos de algumas destas ferramentas, presentes nos sistemas CAD Solid Works e Micro Station/J Modeler.

Estas ferramentas podem ser divididas em dois grupos: *Features* Básicas e *Features* Adicionais. As *Features* Básicas são usadas para gerar as formas iniciais no processo de modelamento do produto, através de operações para modelamento sólido do tipo varredura tais como: projeção ou revolução. As Features Adicionais são usadas para

criar detalhes de projeto nas formas iniciais, através da combinação de operações de varredura com operações booleanas de adição, subtração ou interseção de volumes.



(a) *Features* do MicroStation/J Modeller V7.1 da Bentley

(b) *Features* do Solid Works 99

Figura 3.7: Ferramentas baseadas em conceitos de *features* de sistema CAD comerciais

Cada um destes grupos pode ainda ser dividido em *Features* Genéricas e Específicas. As Genéricas necessitam da intervenção do usuário para especificar a seção transversal ou longitudinal de sua forma geométrica, usando perfis 2D. São exemplos de Features Genéricas: Corte reto (uma *Feature* Adicional Genérica) e Extrusão (uma Feature Básica Genérica). As *Features* Específicas são especializações das Genéricas. Elas têm o perfil 2D de sua seção transversal ou longitudinal previamente definido e fixado pelo desenvolvedor do sistema CAD, durante sua implementação computacional. São exemplos de Features Específicas: Chanfro (uma Feature Adicional Específica) e Barra circular (uma Feature Básica Específica).

Com a combinação de *Features* Genéricas e perfis 2D, é possível gerar formas personalizadas. Deste modo, estes sistemas CAD atendem às necessidades de modelamento geométrico da maioria das peças usadas em projetos mecânicos. Explorando este recurso, é possível criar bibliotecas de definições reusáveis. No MicroStation/J Modeler V7.1, isso é feito armazenando perfis 2D em arquivos específicos, os quais podem, então, ser acessados a partir da caixa de diálogo de *Features* Genéricas. A Figura 3.8 exemplifica este processo mostrando um perfil retangular a ser usado em conjunto com uma *Feature* Genérica de Corte Reto. No Solid Works, salva-se uma *feature* ou um conjunto de *features* já criadas, em um arquivo também específico, que pode ser referenciado em outros modelos. A Figura 3.9, mostra a instanciação de uma *feature*: Protuberância Cilíndrica com Furo Circular Escareado.

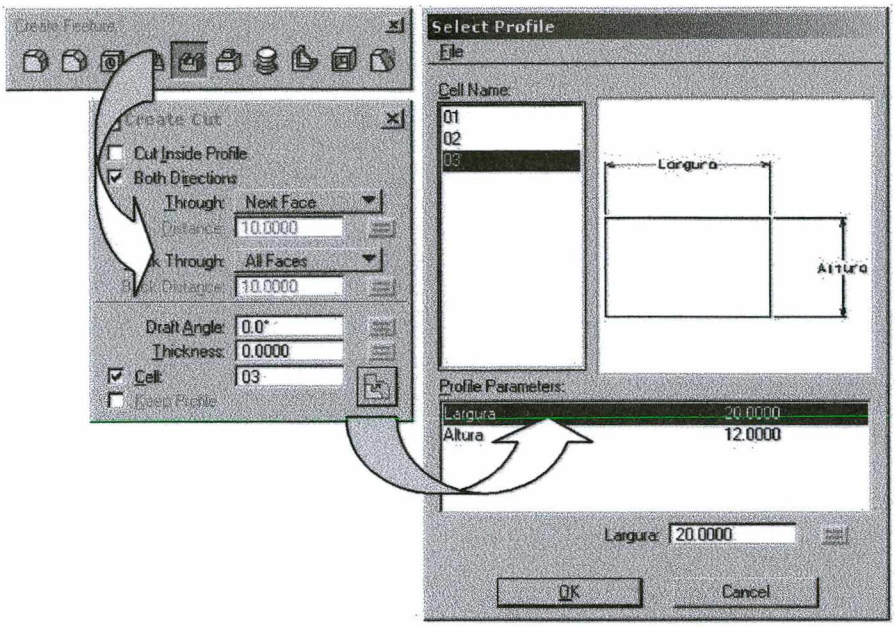


Figura 3.8: Instanciação de uma *feature* personalizada no MicroStation/J Modeler V 7.1

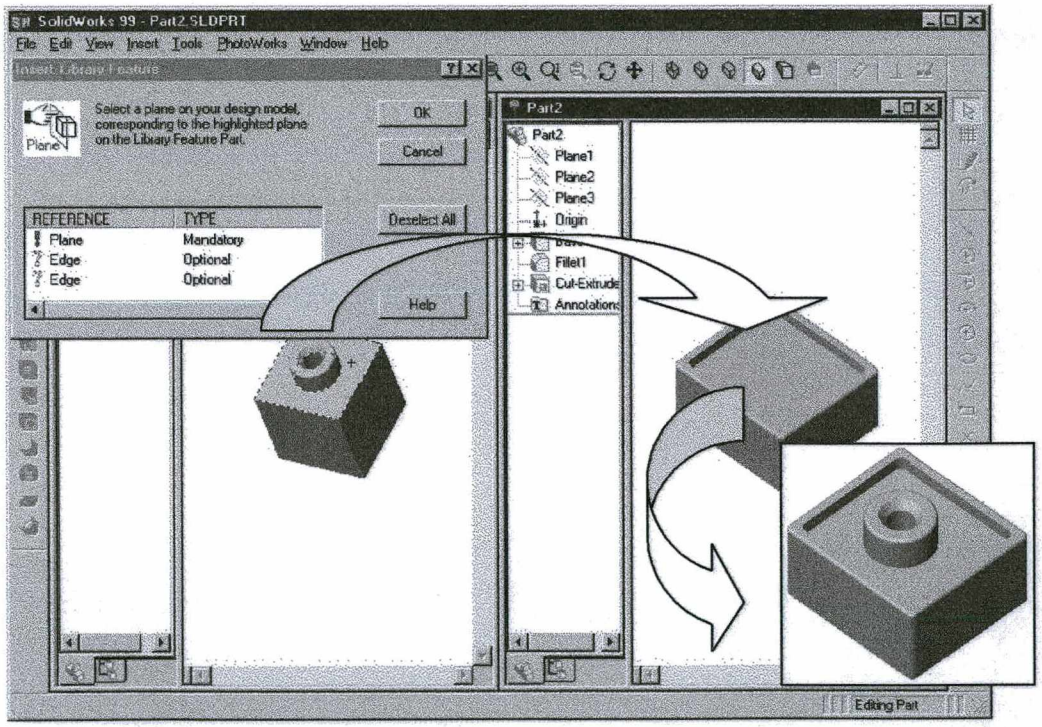


Figura 3.9: Instanciação de uma *feature* personalizada no Solid Works

Existem ainda recursos que permitem o modelamento de restrições de posicionamento entre entidades geométricas e parametrização dimensional. Portanto, é possível gerar perfis 2D parametrizados com relações bem definidas entre as linhas que o constituem, assim como especificar relacionamentos de posicionamento ou dimensionais ente duas ou mais *features*. Deste modo, a propagação automática de alterações no modelo

é efetuada facilmente e quando usados adequadamente, estes recursos podem representar a intenção de projeto no que diz respeito às formas geométricas.

A utilização dos recursos do sistema CAD permite que a representação da intenção de projeto seja viabilizada como mostra a Figura 3.10. No item (a) desta figura, uma ponta de eixo com um rasgo para chaveta deslizante é exibido, junto à árvore de *features* correspondente. No item (b), o diâmetro e comprimento do eixo foram alterados e o rasgo para chaveta transformou-se num vazio no centro do eixo, impossível de ser usinado. No item (c) foi adicionado algumas restrições de relacionamento entre as *features* do modelo, neste caso, as alteração no comprimento e diâmetro do eixo não causaram erros. O rasgo da chaveta continuou a ocupar o posicionamento desejado, de acordo com a intenção de projeto.

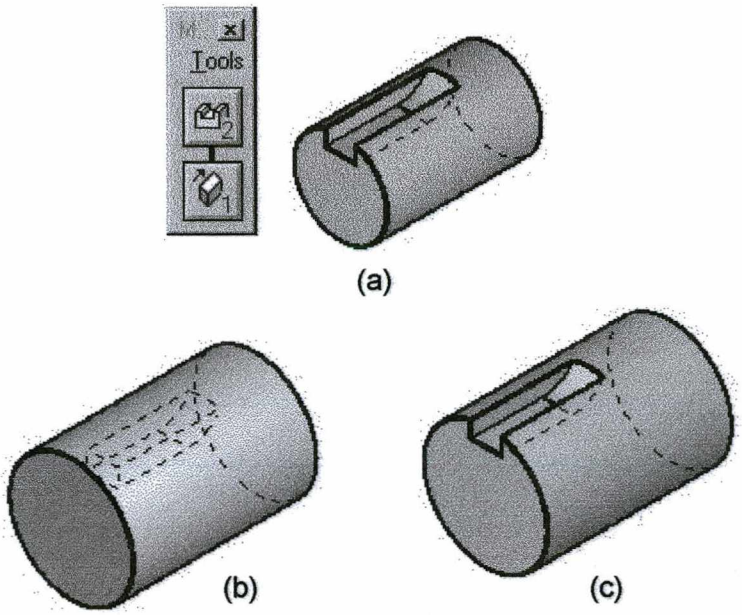
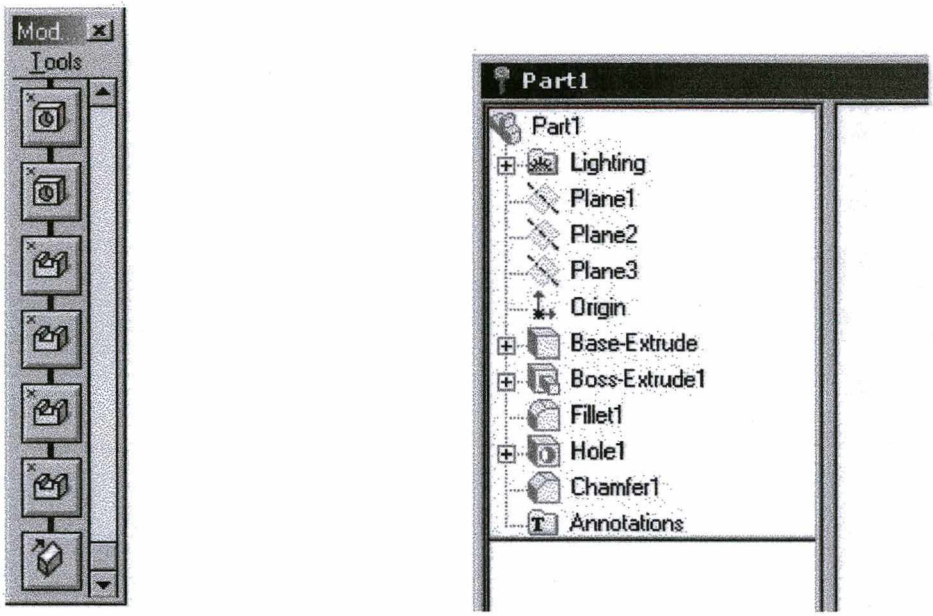


Figura 3.10: Manutenção da intenção de projeto no modelo de *features* utilizando o MicroStation/J Modeler

O processo de modelamento com estas ferramentas é iniciado ao abrir um novo arquivo e criado a primeira *feature* (básica). Continua-se a somar ou subtrair material para alcançar a forma desejada, através da criação de várias *features* adicionais. A ordem ou procedimento de criação é importante, pois uma *feature* não sabe nada sobre as outras criadas anteriormente. Por exemplo, se for criado um furo passante e depois adicionar uma protuberância sobre o furo, ele não será mais passante. Este problema pode ser contornado através da reordenação das *features* manualmente, buscando a manutenção da semântica. Esta reordenação não é livre, pois as *features* adicionais dependem de outras criadas previamente, existe um relacionamento "pai-filho" que deve ser respeitado.

Estruturas denominadas de árvore de *features* ou "históricos do modelo", para as quais são ilustrados exemplos de interfaces de exibição na Figura 3.11, são responsáveis pelo armazenamento do procedimento ou ordem de criação do modelo. Daí vem a denominação dada a sistemas CAD que oferecem esta abordagem de modelamento: "Sistemas de Modelamento Baseados na História". Nesta estrutura, cada nó representa uma *feature* do modelo, que podem ser modificadas pela especificação de novos valores para os seus parâmetros, ou serem eliminadas. Após qualquer modificação, cada operação de modelamento armazenada no histórico é executada novamente, de forma seqüencial (BIDARRA e BRONSVOORT, 2000).



(a) Histórico do modelo no Micro Station/J V7.1 (b)Histórico do modelo no Solid Works 99

Figura 3.11: Exemplos de interfaces que exibem a estrutura de histórico do modelo

3.5 MIGRAÇÃO DOS SISTEMAS CAD CONVENCIONAIS PARA OS BASEADOS EM *FEATURES*

As ferramentas baseadas na tecnologia de *features* presentes em sistemas CAD comerciais de médio ou grande porte, demonstram uma grande evolução nas técnicas de modelamento sólido. Elas são capazes de criar modelos geométricos que representam a intenção de projeto, através da parametrização e estabelecimento de relacionamentos geométricos. Contudo, conforme análise apresentada por BIDARRA, R. e BRONSVOORT, W. F. (2000), estes sistemas CAD apresentam problemas como: elevado custo computacional, limitações de dimensionamento e modelamento devido à forte dependência cronológica de criação das *features* e ao uso de restrições unidirecionais, limitação na

representação e manutenção da semântica das *features*. Deste modo, eles não podem ser considerados como modeladores baseados em *features* genuínos.

Apesar de tantos anos de estudos e experimentações que comprovam a validade da tecnologia de *features*, a ponto de ser reconhecida como o próximo passo para modelamento em sistemas CAD, ainda existem poucas ou limitadas aplicações comerciais desta tecnologia. A dificuldade de sua implementação prática está relacionada à necessidade de troca de dados entre diferentes sistemas computacionais presentes no ambiente de projeto. Este aspecto, que não está totalmente resolvido mesmo para os modelos puramente geométricos, torna-se ainda mais complexo quando a base de dados gerada é fundamentada em definições de *features*, com um volume maior de informações encapsuladas.

Para solucionar este problema, os desenvolvedores de sistemas CAD têm adotado a técnica de programação orientada ao objeto. Assim, todas as entidades de modelamento são representadas através de objetos, com parâmetros e métodos que definem seu "comportamento" e "identidade". Estes sistemas CAD são comercialmente chamados de modeladores de objetos e quando estes "objetos" representam "*features*", como no caso de sistemas CAD de médio ou grande porte, os dois termos acabam se confundindo[5].

Para a troca ou transferência de objetos entre dois sistemas computacionais diferentes, o primeiro problema a ser superado é a limitação imposta pelo uso dos arquivos de dados. Isso porque os arquivos não contêm a definição de objeto, mas apenas seus parâmetros. O sistema que ler o arquivo sem a definição dos objetos implementada internamente, perderá a "inteligência" do mesmo ou simplesmente não o reconhecerá. Diante deste problema, algumas empresas se empenham para desenvolver grupos de objetos padrões a ser implementados por diferentes desenvolvedores de sistemas CAD. Contudo, devido ao grande número de objetos que podem ser necessários para as mais variadas aplicações, o progresso neste sentido tem sido muito lento e alcançado resultados insatisfatórios.

O que se percebe atualmente é que os desenvolvedores de sistemas CAD irão introduzir um novo tipo de arquivo de dados, que armazenará os parâmetros dos objetos constituintes do modelo e aplicativos, tais como os plug-ins da Internet, necessários para sua interpretação. A gerência destes dados torna-se outro problema, a qual, por sua vez, tem sido resolvida através da implementação de um sistema de gerenciamento e

[5] O termo "objeto", pode ter muitas conotações, assim alguns desenvolvedores de sistemas CAD preferem não usá-lo. Nos novos projetos da Bentley por exemplo, as entidades de modelamento são chamadas de "componentes".

armazenamento centralizado, que permite livre acesso e edição de múltiplos usuários[6]. A centralização destas informações facilitará sua manutenção e compartilhamento, criando uma base para desenvolvimento simultâneo do produto.

A resolução destes problemas parece ser o primeiro passo para a implementação de sistemas modeladores de *features* genuínos. Todas as empresas procuram implementar soluções semelhantes. O modo com que irão fazê-lo é que irá diferir.

3.6 CONCLUSÕES

Fica claro o potencial estabelecido pela tecnologia de *features* para desenvolvimento de ambientes computacionais integrados, com interfaces amigáveis e para a implementação de algoritmos que auxiliem as tomadas de decisão, qualquer que seja a atividade de engenharia. Contudo, as *features* são extremamente dependentes da aplicação exigindo um elevado grau de personalização dos sistemas modeladores. Assim, o uso desta tecnologia pode ser mais lucrativo para empresas que fabricam famílias de produtos.

Os estudos teóricos sobre a tecnologia de *features* estão bem evoluídos. Falta a implementação de sistemas computacionais completos capazes de criar modelos válidos que satisfaçam a necessidade de integração proposta pela tecnologia. O maior desafio é permitir que o usuário crie definições personalizadas totalmente integradas com o sistema modelador e que também possam ser igualmente interpretadas por outras aplicações.

Tais questões têm sido resolvidas com sucesso em sistemas modeladores de *features* acadêmicos, já os comerciais ainda deixam muito a desejar. Os sistemas modeladores de *features* comerciais, empregam as técnicas de projeto por *feature* e criação interativa de maneira híbrida. A forma geométrica pode ser armazenada em uma biblioteca, sendo representada implicitamente pela combinação de perfis 2D com operações de varredura e booleanas. A semântica é representada com relacionamentos geométricos definidos manualmente. Contudo, as definições de *features* não encapsulam seus dados geométricos e semânticos em uma única entidade de alto nível. Além disso, não há recursos avançados para a manutenção da validade do modelo, de modo que o usuário deve fazê-lo manualmente.

Todavia, o desenvolvimento das aplicações comerciais é impulsionado pelas demandas de mercado. O momento atual é de transição. O projeto de produtos complexos que integram informações de diversas áreas de conhecimento, criam novas e desafiadoras demandas que impulsionarão a aplicação comercial da tecnologia de *features*. É função dos

[6] O "Project Bank" da Bentley, lançado em 1999, é o primeiro software comercial com esta função.

profissionais especializados nesta área de conhecimento, propor meios para que isso ocorra, em virtude dos benefícios advindos do seu uso.

4 SISTEMAS DE GERENCIAMENTO DE BANCO DE DADOS

4.1 INTRODUÇÃO

Os primeiros aplicativos computacionais tinham seu uso restrito à algumas poucas áreas de atividade e os dados gerados por cada um eram armazenados em arquivos específicos. Na medida em que tais aplicativos foram se tornando mais diversificados e abrangentes, eles passaram a trabalhar com informações mais complexas, muitas vezes de interesse em diversas áreas de atividade e que portanto, deveriam ser compartilhadas. Neste contexto o uso de arquivos específicos passou a ser a fonte de vários problemas tais como: duplicação de dados, retrabalhos (principalmente quando se fazia necessário a expansão dos dados ou de funcionalidades do aplicativo), inconsistência de informações, etc.

Para superar estes problemas surge a proposta de sistemas de gerenciamento de banco de dados (DBMS - *Database Management System*). Um DBMS é basicamente um sistema de manutenção de registros por computador, com objetivo global de disponibilizar um ambiente eficiente para manter informações e troná-las disponíveis quando solicitadas, escondendo do usuário os detalhes de como os dados são armazenados. Ele consiste de uma coleção de dados interrelacionados e estáveis (DB – *Database*), e um grupo de programas usados para acessar, atualizar, e gerenciar estes dados (MS – *Management System*). É necessário a definição formal de estruturas ou modelos de dados, provisão de mecanismos para manipular e garantir a segurança da informação, controles para acesso simultâneo, especialmente se os dados são compartilhados por vários usuários.

Os DBMS podem ser embregados por vários níveis de usuários. ZAIANE, O. R. e KOPERSKI, K (1998) classificam tais usuários em: programadores de aplicação, sofisticados, especializados e nativos. Programadores de aplicação podem utilizá-los para o estabelecimento de uma base dados comum acessada por vários programas através da DML utilizando funções escritas em C/C++, Delphi, Java, etc. Usuários especializados podem modelar aplicações dedicadas utilizando os recursos do sistema. Usuários sofisticados interagem com o DBMS sem escrever programas, através de consultas escritas na linguagem disponibilizada no sistema. Usuários nativos o são aqueles que interagem com o sistema usando as aplicações, sem qualquer tipo de interferência ao que já foi implementado.

As abordagens para implementação prática dos DBMS foram evoluindo ao longo dos anos. Existem atualmente várias soluções comerciais particulares, todavia, baseadas padrões conceituais bem estabelecidos. Neste capítulo é apresentado uma revisão de características genéricas dos banco de dados no que diz respeito à sua arquitetura organizacional e aos diferentes modelos existentes. É dado um maior enfoque aos modelos de dados relacionais, uma vez que esses modelos são usados no presente trabalho. O objetivo é fornecer uma visão introdutória de alguns conceitos básicos relacionados ao assunto.

4.2 ARQUITETURA GERAL DE UM SISTEMA DE GERENCIAMENTO DE BANCO DE DADOS

A preocupação com eficiência dos sistemas de banco de dados, torna necessário o uso de estruturas de dados complexas para fazer o armazenamento dos dados no banco de dados. Esta complexidade deve ser escondida do usuário. Para tal o ANSI/SPARC *Study Group on Database Management System* propõe uma arquitetura de três níveis de abstração, conforme ilustra a Figura 4.1. Esta arquitetura não é a única possível, contudo se adéqua à maioria das abordagens para implementação de um DBMS (DANTE, 1991).

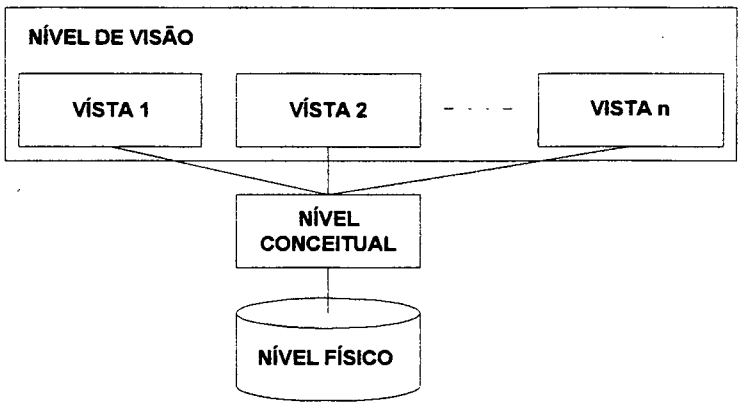


Figura 4.1. Os três níveis de abstração de dados

- Nível Físico: é o nível onde se estabelece como os dados serão armazenados nos arquivos, sendo muito dependente de características do *hardware*.
- Nível Conceituai: este nível é usado pelos administradores do banco de dados, responsáveis pela decisão do conteúdo das informações, estrutura de armazenamento e estratégia de acesso, definição de controles de segurança e integridade, estratégias

de reserva e recuperação, monitoração de desempenho e atender necessidades de modificações.

- Nível de Visão: é o nível de abstração mais alto. Nele, parte das informações são exibidas para o usuário de acordo com suas necessidades específicas. Um mesmo banco de dados pode ser exibido de diferentes maneiras ou ter inúmeras visões.

No Nível Conceitual os administradores do banco de dados recorrem às "linguagens de definição de dados" (DDL – *Data Definition Language*) ao invés de usarem linguagens de programação que acessam diretamente os arquivos em disco. O administrador especifica a estrutura de armazenamento usada em termos de expressões DDL. Estas expressões são compiladas, resultando em um grupo de tabelas contendo os meta-dados (dados com informações sobre outros dados) chamadas "dicionário de dados". O dicionário de dados é inacessível para o usuário, deste modo os detalhes da implementação dos banco de dados ficam escondidos (CUNHA, 2000).

Para a implementação das estratégias de acesso o administrador do banco de dados recorre às linguagens de manipulação de dados (DML – *Data Manipulation Language*). A DML disponibiliza um meio eficiente para recuperação, inserção, eliminação e modificação dos dados. O temo DML e "Linguagem de consulta" (Ex.: SQL), são freqüentemente usadas como sinônimos.

4.3 MODELOS DE BANCO DE DADOS

Os modelos de banco de dados são coleções de ferramentas conceituais para a descrição de dados, relacionamentos, semântica e restrições. Existem modelos lógicos e físicos. Os modelos físicos são usados para descrever os dados no Nível Físico, eles não serão revistos pois estão muito fora do escopo deste trabalho. Os modelos lógicos são empregados nos níveis conceituais e de visão, sendo atualmente uma grande área de pesquisa e desenvolvimento (CUNHA, 2000).

Existem basicamente dois grandes grupos de modelos lógicos: Modelos Lógicos Baseados em Registros e Modelos Lógicos Baseados em Objetos.

Os Modelos Lógicos Baseados em Registros são assim denominados porque a base de dados é estruturada em registros de vários tipos, cada um dos quais define um número fixo de campos ou atributos, com tamanhos ou comprimentos também fixos. Os três modelos mais lógicos mais difundidos são (DANTE, 1991), (CUNHA, 2000), (NETO, FURLAN e HIGA, 1988):

- Modelo Relacional - São os modelos mais usados atualmente. Eles representam uma coleção de dados interrelacionados em forma de tabelas ou relações bi-dimensionais.

As linhas das tabelas possuem os dados armazenados, também chamados de registros ou domínios. As colunas são campos ou atributos que especificam a denominação e o tipo destes dados. As tabelas estão relacionadas entre si numa correspondência unívoca, através de uma coluna ou combinação de colunas denominadas de chave primária, nas quais a qualquer momento, nenhum par de linhas da tabela contém o mesmo valor. Cinco operações fundamentais baseadas na álgebra relacional são responsáveis pela manipulação do banco de dados (transações): seleção, união, projeção, produto, e diferença. Em adição, pode-se definir combinações em termos das operações de produto e seleção. Assim, dados complexos podem ser modelados com base em simples tabelas. A Figura 4.2 ilustra um modelo de banco de dados relacional.

PONTO	X	Y
1	X1	Y1
2	X2	Y2
3	X3	Y3
4	X4	Y4
5	X5	Y5
6	X6	Y6
7	X7	Y7
8	X8	Y8
RELAÇÃO PONTO		

LINHA	PONTO INICIAL	PONTO FINAL
A	1	4
B	1	2
C	2	3
D	3	4
E	5	6
F	6	7
G	7	8
RELAÇÃO LINHA/CURVA		

SUPERFÍCIE	LINHA	TIPO
1	A	LINHA
	B	LINHA
	C	LINHA
	D	LINHA
2	E	LINHA
	F	LINHA
	G	LINHA
	D	LINHA
RELAÇÃO LINHA/CURVA		

Figura 4.2: Exemplo de banco de dados relacional (CUNHA, 2000).

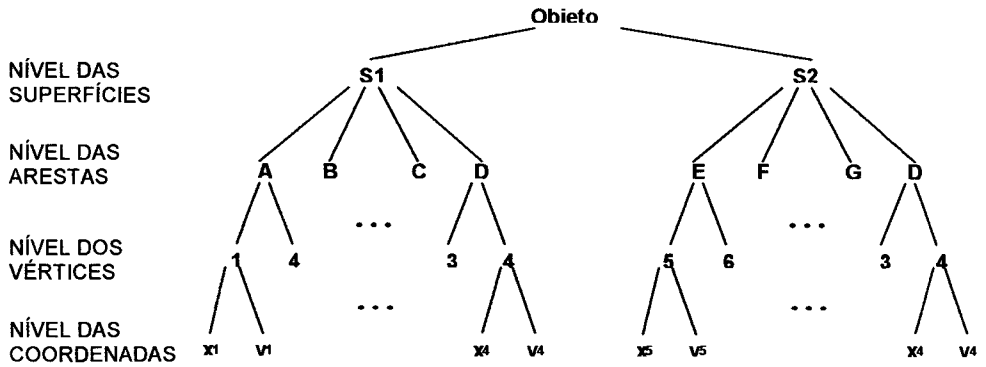


Figura 4.3: Exemplo de banco de dados hierárquico (CUNHA, 2000)

- Modelo Hierárquico - Abordagem dominante na década de 60, são representados por estruturas em árvore cujos elementos, chamados de nós, estão dispostos de maneira hierárquica através de ligações únicas. O topo da estrutura, comumente chamado de raiz, é o objeto modelado através da combinação dos vários nós. Esta estrutura é rápida e fácil mas poucos elementos no mundo podem ser modelados de maneira puramente hierárquica, em adição, a implementação hierárquica usualmente cria

redundâncias e inconsistências. A Figura 4.3 ilustra um modelo hierárquico de banco de dados.

- Modelo em Rede - Desenvolvidos na década de 70, são mais genéricos, pois cada elemento pode ser ligado a vários outros elementos de maneira direta, sendo portando menos restritivos que a estrutura em árvore. A desvantagem está em sua complexidade tanto a nível da estrutura de dados quanto aos algoritmos associados. A Figura 4.4 ilustra um modelo de banco de dados em rede.

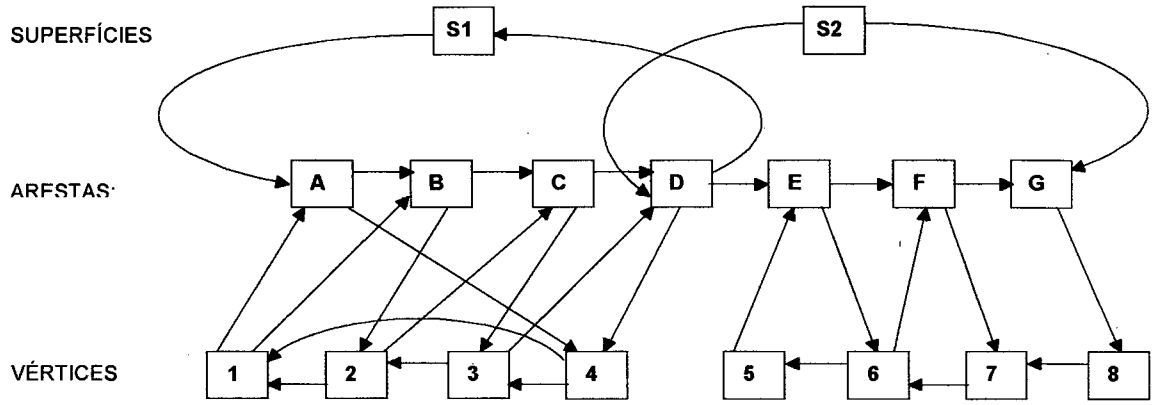


Figura 4.4: Exemplo de banco de dados em rede (CUNHA, 2000)

A proposta dos Modelos Lógicos Baseados em Objetos é relativamente mais recentes que os baseados em registros. Ela surgiu com a necessidade de se construir banco de dados com melhores capacidades de descrição de aspectos semânticos das informações armazenadas. Os dois principais componentes deste grupo são (DANTE, 1991), (CUNHA, 2000):

- Modelo Entidade–Relacionamento - Este modelo se baseia na percepção do mundo como uma coleção de entidades e relacionamentos. Uma entidade é um objeto existente, físico ou abstrato, que pode ser descrito por uma série de atributos associados. O relacionamento é a representação da associação existente entre várias entidades, ao qual também podem estar associados atributos descritivos, de modo que várias regras para controles de integridade podem ser automaticamente estabelecidos. A estrutura lógica dos Modelos Entidade-Relacionamento podem ser expressa graficamente por um "Diagrama E-R", conforme ilustra a Figura 4.5. Os retângulos representam os grupos de entidades, as elipses representam os atributos, os losangos representam relacionamentos entre os grupos de entidades e as linhas fazem a ligação entre estes elementos.

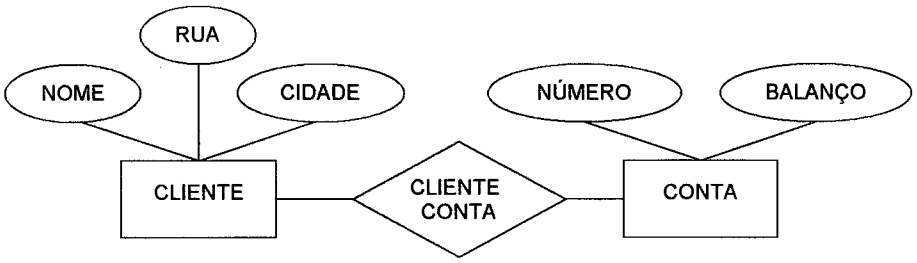


Figura 4.5: Exemplo de um Modelo Entidade - Relacionamento

- Modelo Orientado a Objeto – O processo atual de desenvolvimento das técnicas de modelagem de informação apontam para a intensificação do uso de Modelos Orientado a Objeto. Estes modelos são baseados em uma coleção de objetos com atributos ou variáveis de instância manipuladas através de métodos internos. Eles incorporam os conceitos de herança, abstração, polimorfismo e encapsulamento da técnica programação orientada a objeto. Os objetos se relacionam uns com os outros trocando mensagens através de seus métodos. Aqueles que contêm os mesmos tipos de variáveis de instância e métodos são agrupados em classes, as quais podem ser vistas como o tipo de definição do objeto ou um tipo de dado abstrato. É possível organizar o modelo em hierarquias de classes com vários níveis de abstração e combinar objetos para definir outros mais complexos. A existência de métodos e atributos internos permite o encapsulamento de comportamentos complexos com semânticas detalhadamente definidas, que não são facilmente representados através de simples relações.

4.4 SISTEMA GERENCIADOR DO BANCO DE DADOS

O gerenciador do banco de dados é um módulo do DBMS que disponibiliza uma interface entre os dados de baixo nível e os aplicativos e consultas submetidas ao banco de dados. Seu objetivo fundamental é facilitar e simplificar o acesso aos dados.

Este módulo é basicamente responsável por (ZAIANE e KOPERSKI, 1998):

- Interação com o gerenciador de arquivo, armazenando os dados no disco através do sistema de arquivos usualmente disponibilizado pelo sistema operacional. O gerenciador do banco de dados deve traduzir as declarações DML em comandos de baixo nível do sistema de arquivos.
- Integridade, garantido que as atualizações na base de dados não violam a consistência.
- Segurança, assegurando que os usuários tenham acesso somente aos dados permitidos.

- Cópias de segurança e recuperação, detectando falhas devido à quedas de energia, danos no disco de memória, erros de software, etc.
- Controle concorrente (para os grandes sistemas), preservando a consistência dos dados quando estão sendo acessados editados por múltiplos usuários.

4.5 PROJETO DE BANCO DE DADOS RELACIONAIS

Conforme estabelecido inicialmente, o presente trabalho propõe a programação de uma aplicação que utiliza recursos de um sistema de gerenciamento de banco de dados relacional, deste modo justifica-se uma revisão das técnicas formais para construção de modelos de dados relacionais.

Muitos bancos de dados relacionais são tão simples e trabalham com um número tão limitado de dados que o projetista pode construí-lo de modo satisfatório sem maiores preocupações com a estrutura e os métodos de acesso (VIESCAS, 1995). Contudo, no projeto de um banco de dados relacional é aconselhável, ou para os casos mais complexos, estritamente necessário, estabelecer um bom esquema de relacionamento entre os dados envolvidos. Um mal projeto pode ocasionar problemas como a repetição de dados ou a inabilidade de representar certas informações.

Tal projeto inicia-se com uma análise, determinando as tarefas que deverão ser realizadas pelo banco de dados. Em seguida deve ser feita uma listagem, para cada tarefa, de todos os atributos necessários para sua execução. Então, é preciso organizar estes atributos por assuntos ou tuplas[7] e mapea-los em tabelas ou entidades no banco de dados. O maior desafio está em determinar o modo de organização destas informações, ou seja, definir quais são as relações necessárias e quais devem ser seus atributos. Para facilitar ou conduzir esta tarefa, recomenda-se o uso da técnica de normalização.

A normalização é um processo destinado à organização do banco de dados relacional, de modo que se tenha como resultado uma estruturação de informações não ambígua, sem redundância e desperdício de espaço. Ela consiste de uma série etapas seqüenciais em que se busca o refinamento da estrutura de modelagem dos dados, iniciado depois de identificar os dados que devem estar no banco de dados e as suas relações, ou seja, iniciado após se ter um primeiro esboço do banco de dados.

O processo de normalização não é descrito de forma unânime pelos diversos pesquisadores do assunto. A principal diferença está no grau de refinamento conseguido, o

[7] Uma tupla é um grupo ordenado de dados, podendo ser de tipos variados, geralmente separados entre si por vírgulas e delimitados por parênteses ou colchetes (Ex.: [Alvino,21,10,1975]). Em projetos de bancos de dados relacionais, as tuplas são usadas para representar grupos de registros relacionados e interdependentes. Geralmente correspondem à linha de uma tabela.

qual varia de acordo com o número de etapas constituintes. DANTE, C. J. (1991) apresenta o processo através de seis formas normais, conforme ilustra a Figura 4.6. São elas: Primeira Forma Normal (1FN), Segunda Forma Normal (2FN), Terceira Forma Normal (3FN), Forma Normal de Boyce-Cood (FNBC), Quarta Forma Normal (4FN) e Quinta Forma Normal (5FN) ou Forma Normal Projeção - Junção (FN/PJ). A 5FN é a forma normalizada mais refinada ou final.

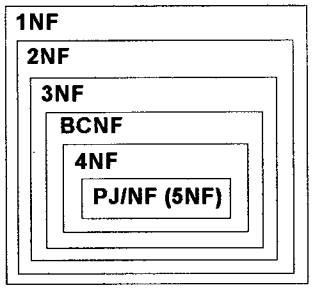


Figura 4.6: Formas normalizadas (DANTE, 1991)

Contudo, o refinamento até a 4FN é suficiente para a maioria dos casos. Assim, nos próximos parágrafos o processo de normalização será descrito em quatro passos, usando os procedimentos padrões sugeridos por NETO, A. F., FURLAN, J. D. e HIGA, W. (1988).

O primeiro passo consiste em submeter ou reduzir as relações existentes neste primeiro esboço do banco de dados à sua Primeira Forma Normal, através dos seguintes procedimentos:

- verificar se há ocorrências repetitivas de atributos dentro da tupla "A" analisada;
- destacar os atributos repetitivos, criando uma nova tupla "B", que absorverá esse itens e herdará também a chave primária da tupla "A";
- estabelecer o relacionamento natural entre "A" e "B".

O segundo passo é a redução das tabelas na Primeira Forma Normal à sua Segunda Forma Normal. Para isso, deve-se:

- verificar se a tupla "C" analisada necessita de dois ou mais atributos para ser individualizada (possui chave concatenada);
- destacar os atributos que dependem parcialmente da chave concatenada, criando um a nova tupla "D", que absorverá estes itens e a chave primária componente da tupla "C";
- estabelecer o relacionamento natural entre "C" e "D".

Uma relação está na Segunda Forma Normal se, e somente se, estiver na Primeira Forma Normal, e cada atributo "não-chave" for totalmente dependente de sua chave primária.

No terceiro passo, as relações obtidas no passo anterior devem ser submetidas à Terceira Forma Normal, sendo atingida pelas seguintes providências:

- verificar se a tupla "E" analisada possui atributos que são dependentes de outros atributos também contidos nela. Esta relação de dependência pode ser com um atributo herdado de uma entidade relacionada onde era chave primária (chave estrangeira) ou com atributos que são resultantes de algum cálculo específico;
- destacar os atributos dependentes da chave estrangeira e incorporá-los na tupla "F", de onde foi herdada a referida chave;
- eliminar os atributos obtidos por cálculo a partir de outros atributos.

Uma relação está na Terceira Forma Normal se, e apenas se, os atributos "não-chave" desta relação forem mutuamente independentes, e totalmente dependentes da chave primária da relação.

No quarto passo, as relações obtidas no passo anterior devem ser submetidas à Quarta Forma Normal, através do seguinte roteiro:

- verificar se a tupla "G" analisada possui atributos "não-chave" multivalorados e independentes, associados ao mesmo valor da chave;
- destacar os atributos "não-chave" multivalorados, criando novas tuplas individualizadas para cada um deles, residentes em entidades distintas, herdando também a chave original.

Coimo descrito, alguns autores tais como DANTE, C. J. (1991), sugerem a extensão do processo de normalização adicionando a Forma Normal de Boyce-Cood (uma melhoria da Terceira Forma Normal) e a Quinta Forma Normal ou Forma Normalizada Projeção-Junção. Deste modo pode-se conseguir um melhor refinamento das relações. Contudo, o próprio Dante, C. J. admite em sua obra que não há muito rigor no processo de normalização de banco de dados relacionais, e que de fato, este processo é essencialmente um conjunto de idéias simples e de bom senso (DANTE, 1991, pg 378).

4.6 CONCLUSÕES

Neste capítulo foi feita uma revisão de conceitos gerais sobre sistemas de gerenciamento de banco de dados. Tais sistemas são normalmente implementados com base em uma arquitetura de três níveis de abstração e podem ser divididos em dois módulos, o sistema de gerenciamento e a base ou banco de dados, a qual possui a estrutura formal ou modelo de representação de dados. Foi dado um maior enfoque no aspectos relacionado ao projeto de modelos de dados relacionais, já que estes serão usados no presente trabalho.

Todas as abordagens de implementação de banco de dados são bastante genéricas. Normalmente um mesmo problema pode ser resolvido com qualquer modelo de banco de dados. O administrador fica restrito basicamente à disponibilidade de ferramentas e conhecimento técnico. De um modo geral, os modelos de dados procuram representar os dados em níveis elevados de abstração de fácil entendimento, respeitando a semântica inerente à informação modelada. Neste sentido os Modelos Lógicos Baseados em Objetos são os mais evoluídos.

5 PROPOSIÇÃO DE UMA ABORDAGEM PARA EXPLORAÇÃO DA TECNOLOGIA DE *FEATURES* EM SISTEMAS CAD COMERCIAIS EM CONJUNTO COM RDBMS

5.1 INTRODUÇÃO

Este capítulo descreve com maiores detalhes a proposta do presente trabalho: uma abordagem para exploração da tecnologia de *features* já disponível em sistemas CAD comerciais em conjunto com RDBMS, para modelamento de peças. O objetivo é esclarecer os aspectos mais relevantes para sua implementação prática.

5.2 A ABORDAGEM PROPOSTA

A abordagem se baseia na integração CAD-RDBMS, para construção e uso de uma biblioteca de *features* personalizadas a serem utilizadas no detalhamento de projetos. As definições personalizadas de *features* são entendidas como unidades informacionais de projeto reusáveis. Elas são constituídas por entidades geométricas e não geométricas. As entidades geométricas são representadas com recursos disponíveis no sistema CAD, as não geométricas são parâmetros dimensionais da forma geométrica e atributos tecnológicos tais como: peso, material, processo de fabricação, etc., representados em um banco de dados relacional. Essa idéia é mostrada na Figura 5.1.

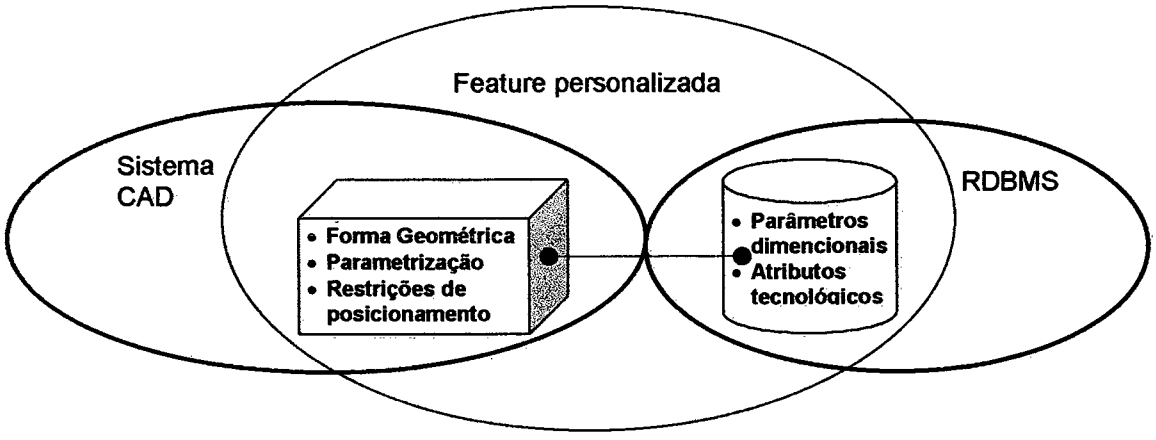


Figura 5.1: *Feature* personalizada

Um ambiente para modelamento destas unidades informacionais pode ser implementado na forma de um aplicativo computacional integrado a um sistema CAD, cujo funcionamento geral é ilustrado na Figura 5.2. O banco de dados: "BD reusáveis", é usado para auxiliar o modelamento das *features* personalizadas, ele armazena dados não geométricos padronizados. Tais dados são estabelecidos em trabalhos anteriores, realizados por diferentes áreas de conhecimento presentes no sistema de manufatura.. A ferramenta computacional, denominada: "Interface CAD-RDBMS", possibilita que o usuário visualize e reuse dados padronizados do "BD reusáveis", ou defina outros personalizados, transfira os parâmetros dimensionais para geração das formas geométricas no CAD e armazene estes parâmetros em conjunto com os atributos tecnológicos relacionados, no "BD reusados". O "BD reusados", registra a porção não geométrica das *features* personalizadas, a qual contém informações úteis para novos projetos, reprojetos ou aplicações subsequentes

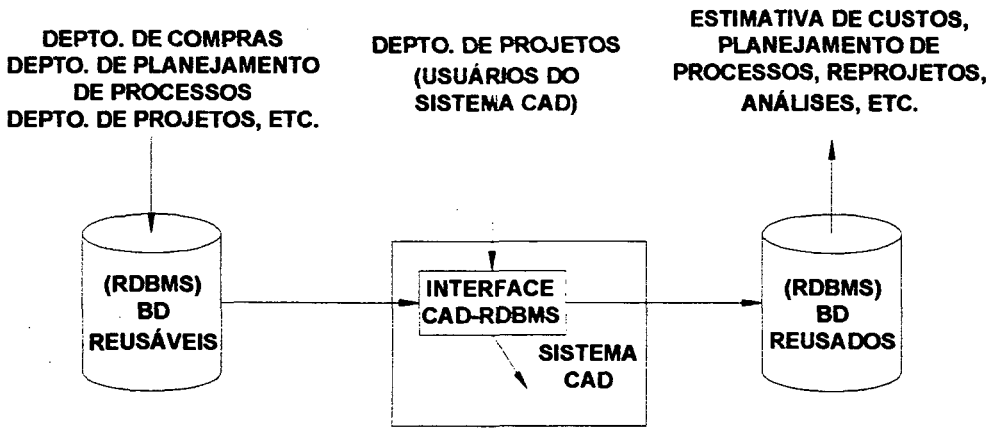


Figura 5.2: Esquema geral de funcionamento do aplicativo computacional para implementação da proposta

Neste trabalho, não há preocupação em se desenvolver um novo modelo de dados do produto. Conforme descrito, a proposta enfoca os aspectos relacionados à integração de sistemas computacionais. O modelo do produto é composto pelo arquivo de dados do sistema CAD e o banco de dados relacional "BD reusados".

5.3 DETALHAMENTO

Para implementação da proposta deve-se estabelecer os esquemas de representação das informações de projeto no RDBMS e no sistema CAD, assim como uma forma de comunicação entre eles. Os dois esquemas de dados devem ser estruturados em

termos de definições de *features* personalizadas, facilitar a expansão da biblioteca de dados e ser independentes de um sistema CAD ou RDBMS específico.

5.3.1 Informações de projeto no RDBMS

Para representação das informações de projeto no banco de dados "BD reusáveis", elas são organizadas em grupos independentes de tabelas, conforme o esquema apresentado na Figura 5.3. Em cada grupo existe uma tabela principal, com dados que se referem a uma *feature* personalizada específica. Esta tabela registra atributos tecnológicos escolhidos para abstrair outros conjuntos de dados não geométricos armazenados em tabelas secundárias. Cada linha da tabela principal se relaciona com várias linhas de uma única tabela secundária. Na Figura 5.3, a tabela principal se refere a uma *feature* chamada "Barra Retangular". Ela está relacionada com duas outras tabelas que armazenam conjuntos de dados não geométricos diferenciados pelo atributo material, em: Aço e Latão. Nas consultas a este grupo de tabelas, o usuário especifica o material da *feature* e então escolhe os parâmetros dimensionais normalizados, no caso: largura e espessura, assim como o atributo tecnológico, no caso: peso.

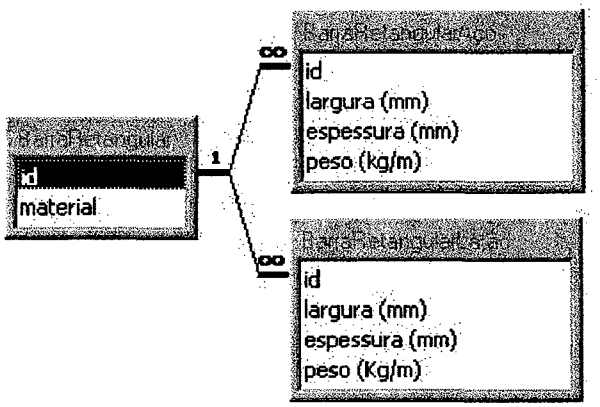


Figura 5.3: Esquema para representação de dados não geométricos padronizados no "BD reusáveis"

Para representação das informações de projeto "BD reusados", ele é organizado com um grupo único de três tabelas conforme Figura 5.4. Uma tabela principal chamada "*Features*", possui registros que permitem identificar as *features* usadas no modelamento de um produto. Cada linha desta tabela identifica uma única *feature* e se relaciona com várias linhas de duas tabelas secundárias. Uma tabela secundária armazena os atributos tecnológicos e a outra armazena os parâmetros dimensionais.

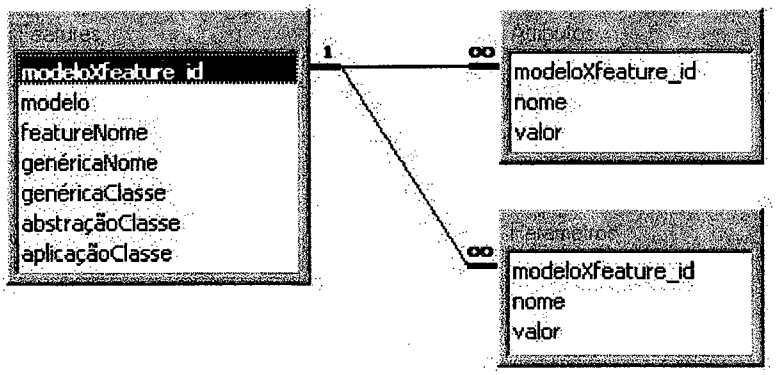


Figura 5.4: Esquema para representação de dados geométricos de projeto reusados para o modelamento de um produto, no "BD reusados"

Cabe aqui uma consideração sobre as informações de projeto no RDBMS. Evidentemente, qualquer modelo de banco de dados poderia ter sido usado para implementar tais informações, uma vez que, por princípio, tais modelos desenvolvidos com propósitos genéricos. Aqui optamos por utilizar um modelo relacional devido ao fácil acesso à ferramentas para gerenciamento deste tipo de modelo (RDBMS) e ao grande volume de usuários do mesmo no meio industrial, comercial e acadêmico.

5.3.2 Informações de projeto no sistema CAD

O esquema de representação das informações de projeto no ambiente CAD tem o objetivo usar as ferramentas para modelamento de *features* presentes nestes sistemas, para modelar os dados geométricos da *feature* personalizada. Tais ferramentas podem ser divididas em quatro grupos: ferramentas para parametrização de perfis 2D, *Features* Genéricas, ferramentas para especificação de restrições de posicionamento geométricos e ferramentas para especificação de variáveis dimensionais globais. Adicionalmente, propõe-se a utilização do API do sistema CAD para implementar algoritmos de automatização do método construtivo de cada *feature* personalizada. Utilizando estes algoritmos, os ajustes específicos das *features* genéricas, atribuição de parâmetros dimensionais, estabelecimento de relações geométricas e variáveis globais serão realizados automaticamente.

A forma de representação computacional destes dados garante o inter-relacionamento. Isso torna possível a implementação de uma interface que trabalhe com todas estas ferramentas, como se fossem uma única entidade de modelamento. A Figura 5.5 ilustra este esquema de representação, o qual é descrito com maiores detalhes nos próximos parágrafos.

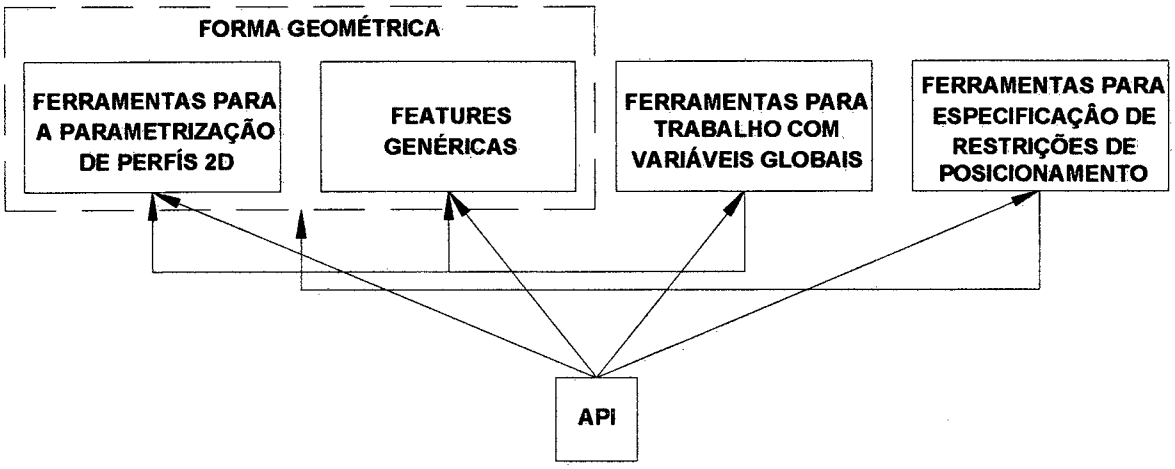


Figura 5.5: Esquema de representação das informações de projeto no sistema CAD

Propriedades de forma e parâmetros de relacionamentos intrínsecos são representados em termos de perfis 2D parametrizados e das *features* Genéricas presentes no sistema CAD. Esta idéia é ilustrada na Figura 5.6, onde um rasgo de chaveta é modelado em uma ponta de eixo utilizando uma *feature* genérica de Corte Reto e o perfil 2D parametrizado de sua seção longitudinal.

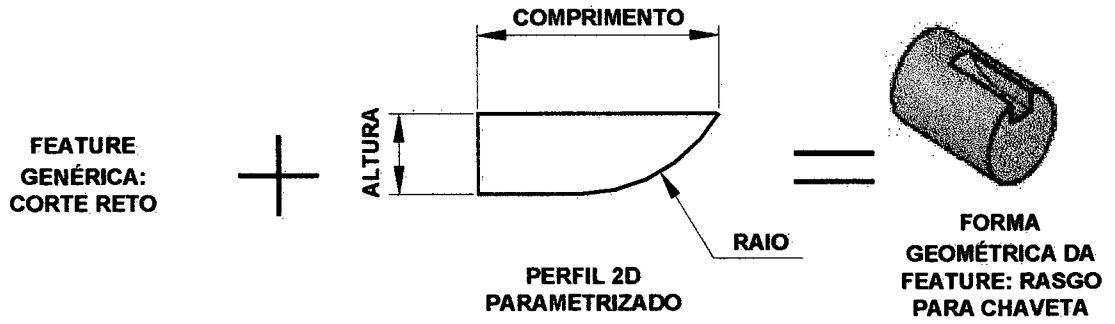


Figura 5.6: *Feature* em termos de uma definição genérica e um perfil parametrizado

As ferramentas para especificação de restrições de posicionamento são usadas para modelar relacionamentos geométricos entre duas ou mais *features*, segundo a intenção de projeto na peça. Ferramentas para especificação de variáveis globais são usadas para representar parâmetros dimensionais dependentes ou derivados, podendo atuar sob os parâmetros dimensionais das *Features* Genéricas e dos perfis 2D parametrizados.

5.3.3 Comunicação CAD-RDBMS

A comunicação entre os sistema CAD e RDBMS é estabelecida por funções que atuam sobre os dois esquemas de representação das informações de projeto. Elas deverão

permitir o fluxo de informação do "BD reusáveis" para as ferramentas do sistemas CAD e da Interface CAD-RDBMS para o "BD reusados".

O modo de implementação de tais funções é dependente dos recursos oferecidos pelo sistema CAD e RDBMS usados e também deverão trabalhar de maneira diferente para cada *feature* personalizada. O que se propõe é uma classificação genérica das variáveis manipuladas por tais funções, com o intuito de facilitar as implementações específicas. As variáveis que podem ser classificadas em:

- Parâmetros construtivos: são as variáveis necessárias para realizar ajustes específicos no método construtivo das ferramentas para modelamento de *features* do sistema CAD, tais como: orientação da construção, nome do perfil 2D, etc.
- Parâmetros dimensionais: se referem às dimensões geométricas da *feature*, e são divididos em derivados, independentes e normalizados. Os parâmetros dimensionais derivados são aqueles que derivam de outras *features* presentes no modelo, os independentes não derivam de nenhum outro, os normalizados derivam de normas. Portanto, cada tipo de variável deve ser processada de forma diferente, sendo que esta classificação é necessária para facilitar a implementação dos métodos que farão isto.
- Atributos tecnológicos: são informações que irão enriquecer o modelo, tais como material e sua densidade, processo de produção e ferramentas usadas, tempo, custo, etc. conforme mencionado no item anterior.

5.4 PROCESSO DE IMPLEMENTAÇÃO DA PROPOSTA

Nesta seção é apresentado o processo de implementação da proposta. O intuito é sistematizar esta atividade, de modo que os aspectos mais relevantes para condução dos trabalhos sejam enfatizados. A Figura 5.7 ilustra este processo, ele é constituído de quatro atividades básicas.

Na primeira etapa é realizado a especificação dos recursos materiais e escopo das informações de projeto. Em outras palavras, esta atividade consiste em determinar quais serão os hardware e software adotados, assim como o escopo das *features* que serão implementadas. Para tal, deve ser escolhido uma linha de produtos específicos e uma empresa interessada em implementar a proposta e idealmente, utilizar os recursos computacionais já disponíveis. Os objetivos almejados pelos futuros usuários devem ser bem esclarecidos. É importante ter em mente que o grupo de *features* a ser implementado só será capaz de satisfazer as necessidades de modelamento das linhas ou classes de produtos específicos escolhidos para sua identificação e formalização.

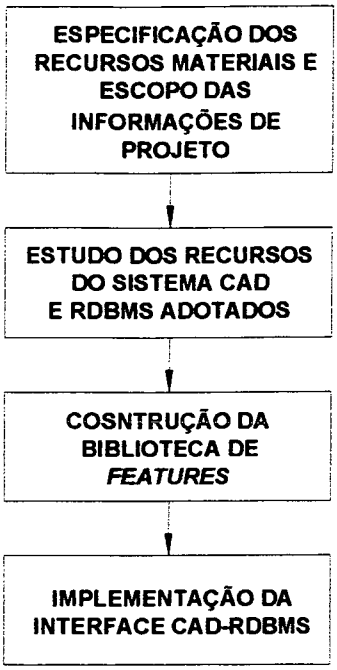


Figura 5.7: Fluxograma do processo de implementação da proposta

A segunda atividade a ser realizada, consiste no estudo dos recursos disponíveis no sistema CAD e RDBMS escolhidos. Deve ser feita uma avaliação técnica das ferramentas para modelamento de *features*, as ferramentas padronizadas para interação entre estes dois sistemas, API oferecido, documentações, etc. Maximizando o uso de recursos já existentes, é possível minimizar a quantidade de algoritmos necessários para implementação da proposta, tornando o processo mais rápido e barato.

A terceira etapa é a construção da biblioteca de *features*. Ela pode ser dividida em três atividades básicas que são:

- identificação e formalização das *features*: feito seguindo a metodologia proposta por SHAH, J. J. e MÄNTYLÄ, M. (1995), a qual foi descrita resumidamente no item 3.3.1;
- armazenamento das definições identificadas e formalizadas no sistema CAD: o resultado desta atividade é dependente dos recursos disponíveis no sistema CAD e devem influenciar fortemente na etapa de implementação da Interface CAD-RDBMS;
- registro das informações não geométricas no RDBMS.

A última atividade compreende a implementação da Interface CAD-RDBMS. Devem ser implementados os algoritmos de interação com os esquemas de representação das informações de projeto e as interfaces de interação com o usuário.

5.5 CONCLUSÕES

Este capítulo propôs uma abordagem de modelamento de produtos com base na integração de recursos de sistemas CAD comerciais de médio e grande porte com RDBMS. Por princípio e em termos genéricos, a proposta permite o reuso de informações geométricas e não geométricas de projeto interrelacionadas, para a criação de modelos de peças. Isso possibilita a previsão de uma série de benefícios, contudo é necessário uma implementação prática da abordagem para comprovação de sua validade e fundamentação de conclusões definitivas ao seu respeito.

6 IMPLEMENTAÇÃO DA PROPOSTA

6.1 INTRODUÇÃO

Neste capítulo é apresentada a experimentação prática da proposta descrita no capítulo anterior. Houve a preocupação de implementar apenas as funcionalidades principais do aplicativo: Interface CAD-RDBMS, de modo a demonstrar seu funcionamento básico. Tais funcionalidades são aquelas necessárias para acessar os dados não geométricos armazenados no banco de dados, usá-los para criar modelos baseados em *features* no sistema CAD e armazenar os dados usados em outro banco de dados relacionai associado ao modelo.

Os resultados irão auxiliar no entendimento da proposta, visualização de seus benefícios e dificuldades de implementação. Eles não são definitivos, mas representam um protótipo ou produto piloto de um aplicativo integrado ao sistema CAD MicroStation/J que deve ser aperfeiçoado para aplicações comerciais.

6.2 ESPECIFICAÇÃO DOS RECURSOS MATERIAIS E ESCOPO DAS INFORMAÇÕES DE PROJETO

Os principais recursos utilizados para a implementação da proposta, foram disponibilizados no laboratório GRANTE, localizado no Bloco A da Faculdade de Engenharia Mecânica da Universidade Federal de Santa Catarina. Tais recursos consistem de:

- PC / Windows, Pentium III 500MHz, 128Mb RAM;
- Sistema CAD MicroStation/J V7.1 da Bentley, com o módulo de configuração para engenharia: Modeler;
- RDBMS Access 97, da Microsoft (MS Access 97);

Também foram utilizados software gratuitos disponibilizados na Internet, são eles:

- GNU Emacs, um editor de texto com algumas funcionalidades e configurações úteis para programação com diversas linguagens, dentre elas: Java e C/C++
- JDK (Java Development Kit), API padrão da linguagem Java fornecida pela Sun Microsystems.

Como recursos bibliográficos:

- manuais do MicroStation/J Modeler e MS Access;
- livros técnicos sobre programação com a linguagem Java;
- livro técnico sobre programação com a linguagem C;

A identificação e formalização das *features*, e consequentemente, seu campo de aplicação, é limitado a peças de produtos fabricadas por empresas do ramo metal mecânico. Para tal, foram adquiridos desenhos de projeto de uma Betoneira 120L, produto fabricado pela Metalúrgica Erwino Menegotti Ltda.. Esta empresa atua na produção de equipamentos para processamento de concreto, tais como betoneiras, fôrmas, etc. Sua sede principal se localiza em Jaraguá do Sul-SC, Brasil. Maiores detalhes sobre a Menegotti e seus produtos podem ser encontrados no site: "http://www.menegotti.ind.br".

6.3 ESTUDO DOS RECURSOS DO SISTEMA CAD E RDBMS ADOTADOS

Os recursos do sistema MicroStation/J Modeler V7.1 da Bentley e MS Access 97 podem ser estudados detalhadamente em livros especializados e manuais do usuário facilmente encontrados em bibliotecas e nos sites dos fabricantes na Internet. A este respeito, tem-se como referências principais: (MICROSTATION Modeler, User Guide, 1996), (SAHAI, 1999), (VIESCAS, 1995). O estudo apresentado a seguir é apenas uma descrição de aspectos considerados como mais relevantes para o entendimento do presente trabalho.

6.3.1 Recursos do sistema CAD

O sistema MicroStation/J Modeler V7.1 da Bentley, possui várias *features* Genéricas e Específicas, Básicas ou Adicionais, junto com recursos para criação de bibliotecas de definições personalizadas e outros para determinação de restrições geométricas e variáveis globais, conforme descrito no item 3.4.2. No presente trabalho, há interesse nas *Features* Genéricas que necessitam de um único perfil 2D, para definição de sua forma geométrica. A Figura 6.1 mostra quais são estas *features*.

Outro recurso importante disponibilizado por este sistema CAD são as ferramentas para ligação de entidades geométricas com banco de dados externos e EEDs. Tais ferramentas poderiam ser usadas para ligar a forma geométrica de uma *feature* personalizada com seus dados não geométricos, através de seu perfil 2D. Contudo, foi verificado que isto não é possível. No caso do banco de dados, isso é devido ao fato de os perfis, ao serem usados para modelar uma *feature*, alterarem sua estrutura de dados interna, perdendo as informações que estabelecem sua ligação com a base de dados

externa. Isso também acontece com as informações representadas por EEDs. Ainda, como agravante os EEDs limitam o método de organização das informações não geométricas.

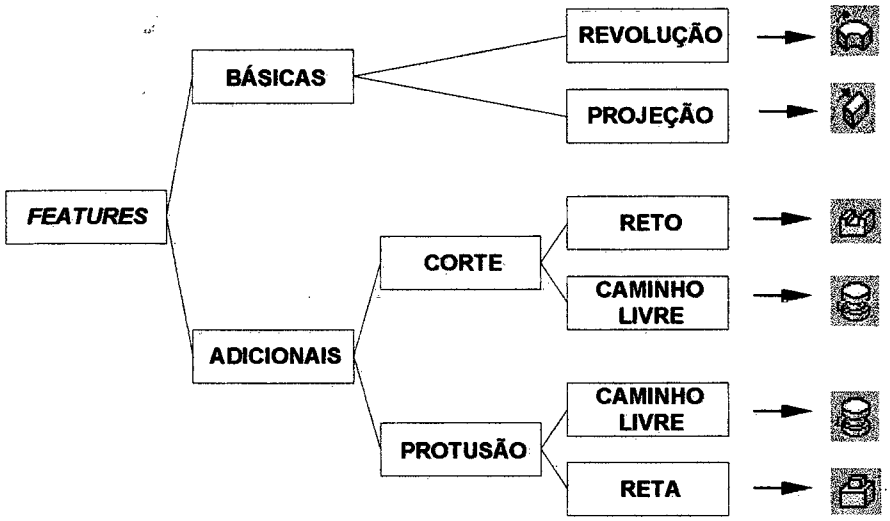


Figura 6.1: *Features* Genéricas de interesse no MicroStation/J Modeler V7.1

O MicroStation/J Modeler V7.1 também possui recursos para servir e receber DDE's. Com eles é possível escrever macros, utilizando a linguagem Basic, que enviam dados do Access para o MicroStation/J, através de comandos de linha denominados: "keyins". Os "keyins" podem ser usados para selecionar qualquer ferramenta, checar ou ajustar configurações. Quando são usados para selecionar ferramentas de modelamento, muitas vezes é possível passar parâmetros construtivos e assim, automatizar a criação de entidades geométricas. Então, seria possível usá-los para transmitir parâmetros construtivos armazenados no Access para as ferramentas de construção de *features* Genéricas do MicroStation/J Modeler. Contudo, a utilização destes recursos também não foi possível, devido ao caráter limitado dos "keyins" específicos para trabalho com as ferramentas do módulo Modeler.

Portanto, para a implementação da proposta, serão necessárias novas soluções, tanto para fazer o relacionamento entre a forma geométrica da *feature* e seus dados não geométricos, quanto para possibilitar a utilização automática dos dados não geométricos no modelamento das *features*. A princípio serão implementados algoritmos personalizados e isso é possível utilizando as linguagens de personalizações avançadas do MicroStation/J, são elas:

- MDL: é um dialeto da linguagem C, usada pelos desenvolvedores do MicroStation e terceiros interessados em personalizações avançadas. Possui recursos para criação de interfaces gráficas de interação com usuários, acesso e manipulação direta da base de dados do sistema, automatização de tarefas, etc.

- JMDL: é uma extensão da linguagem Java da Sun Microsystems. Na versão do MicroStation/J Modeler usada, o JMDL possui, além de classes específicas para programação no ambiente do sistema CAD, o JDK1.1.8 mais o pacote Java Swing 1.1.1. A Bentley introduziu esta linguagem no MicroStation/J com o intuito de fornecer uma ferramenta de programação orientada ao objeto, tão completa quanto a MDL.

Os algoritmos necessários poderiam ser implementados tanto em MDL quanto em JMDL. Contudo, o JMDL é um projeto relativamente novo e ainda incompleto, especialmente com relação aos módulos de configuração de engenharia (Modeler). Assim, decidiu-se tirar o máximo de proveito das técnicas de programação orientada ao objeto e facilidades da linguagem JMDL e quando necessário, recorrer a funções da linguagem MDL.

6.3.2 Recursos do RDBMS

O MS Access 97 é o RDBMS mais utilizado atualmente em aplicações de pequeno porte, tal qual a proposta deste trabalho. Ele possui uma interface bastante amigável e uma série de recursos que facilitam a criação dos modelos relacionais.

Assim como qualquer outro RDBMS o MS Access 97 usa a álgebra relacional para disponibilizar um ambiente para recuperação e armazenamento de informações na base de dados e implementa interfaces com linguagens (DML) de acesso e atualização padronizadas. A interface de acesso mais utilizada atualmente, inclusive pelo Access, é a ODBC (*Open Database Connectivity*), que emprega a linguagem SQL (*Structured Query Language*). Outra interface que vem sendo crescentemente adotada é a JDBC (*Java Database Connectivity*) que utiliza a linguagem Java. Os RDBMSs que adotam a interface ODBC, também podem ser acessados pela linguagem Java, utilizando uma interface extra conhecida como: Ponte JDBC-ODBC (*JDBC-ODBC bridge*).

Para a criação de consultas simples ou complexas, o MS Access 97 fornece uma uma interface gráfica bastante intuitiva que emprega um método denominado *Querry by example* (QBE). Isso elimina a necessidade de conhecimentos a respeito da linhagem SQL, principalmente quando os dados são manipulados apenas no ambiente do RDBMS. Este não é o caso do prosente trabalho, já que os dados serão trocados entre dois ambientes computacionais (sistema CAD e RDBMS), contudo este recurso também foi de grande valia.

6.4 CONSTRUÇÃO DA BIBLIOTECA DE *FEATURES*

Tendo sido feito o estudo dos recursos disponíveis no sistema CAD e RDBMS, o próximo passo é construir uma biblioteca de *features* para possibilitar a implementação do aplicativo: "Interface CAD-RDBMS". Nos próximos itens este processo é apresentado

seguindo suas três etapas básicas: Identificação e formalização das *features*, armazenamento das definições no sistema CAD, registro das informações não geométricas no RDBMS.

6.4.1 Identificação e formalização das *features*

Utilizando a metodologia de identificação e formalização de *features* proposta por SHAH, J. J. e MÄNTYLÄ, M. (1995) e descrita de forma resumida no item 3.3.1, foi construída uma biblioteca de definições de *features* personalizadas. O produto usado para isso foi a Betoneira 120L, ilustrada na Figura 6.2. Ela possui várias peças fabricadas por diferentes processos tais como: fundição, usinagem, conformação, etc. O trabalho dá enfoque às definições de *features* que exercem alguma função de projeto e que podem ser modeladas com um perfil 2D e uma *Feature* Genérica do MicroStation/J Modeler V7.1.

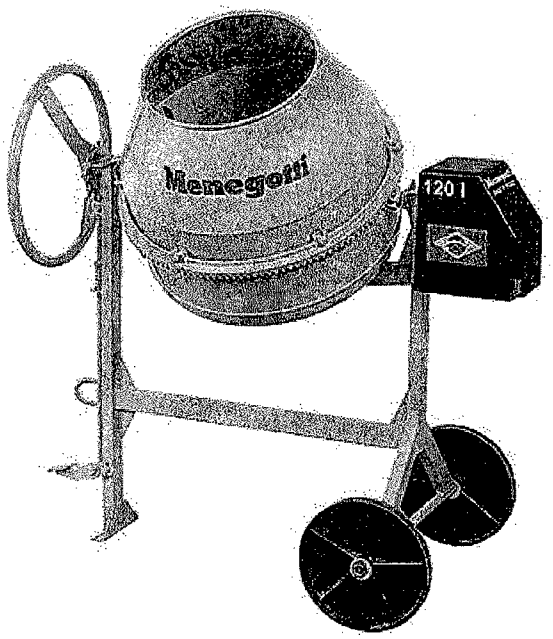


Figura 6.2: Betoneira 120L

A biblioteca completa, com todas as *features* identificadas e formalizadas na Betoneira 120L está no Anexo 1.A seguir a atividade de construção da biblioteca de *features* é exemplificada apenas para uma peça da Betoneira 120L: o Eixo do Pinhão mostrado na Figura 6.3.

Na análise voltada ao produto, primeiro passo do processo de identificação e formalização das *features*, foram identificadas seis formas macro simples que: exercem uma função de projeto, são reusáveis, são parametrizáveis e pertencem ao vocabulário do projetista. Na Figura 6.3, cinco formas macro correspondentes às *Features* Adicionais são evidenciadas.

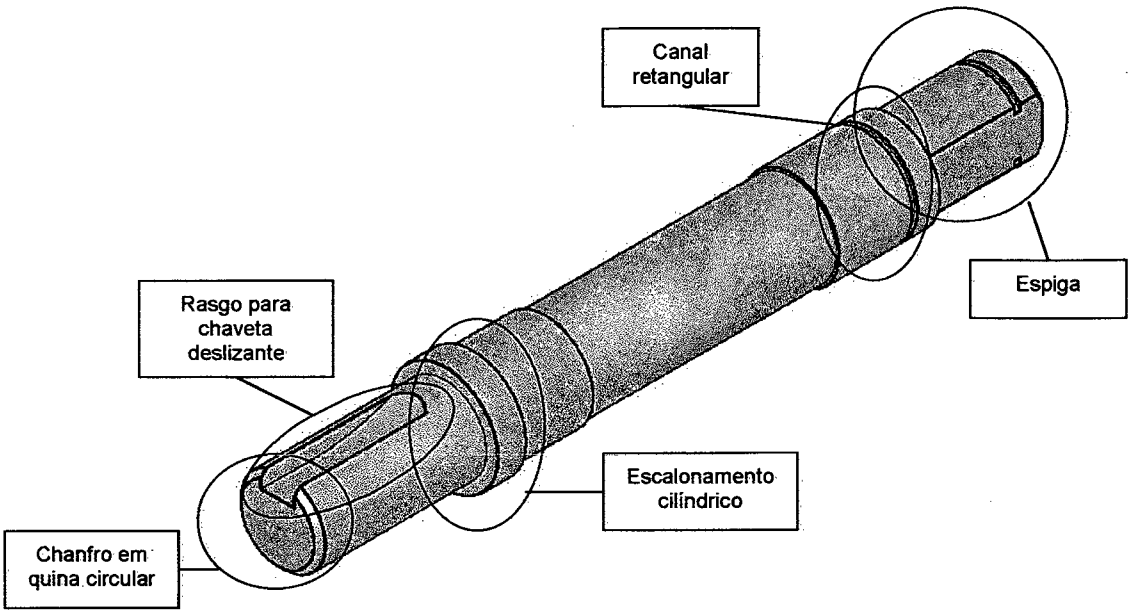


Figura 6.3: Eixo do Pinhão da Betoneira 120L mostrando algumas *features* identificadas

Na análise voltada ao processo de projeto, as *features* são estudadas quanto ao seu processo construtivo. São identificados os parâmetros de entrada para sua definição, relacionamentos intrínsecos necessários para o modelamento da intenção de projeto. Os parâmetros intrínsecos, conforme descrição anterior, são representados de forma implícita através dos perfis parametrizados. Os dados referentes à parâmetros extrínsecos não são identificados por não serem necessários para a implementação das funcionalidades básicas da ferramenta.

As informações obtidas na análise voltada ao produto e ao processo de projeto são agrupadas fazendo um esboço das *features* e suas propriedades, utilizando caneta e papel. As definições encontradas foram avaliadas, validadas e classificadas.

A estrutura de classificação adotada favorece a reusabilidade e rápido acesso às informações. Ela é baseada nas ferramentas usadas para definição de sua forma geométrica, e aspectos intuitivos para o projetista tais como: função e denominação genérica, conforme ilustra a Figura 6.4. Os aspectos funcionais dizem respeito ao que pode ser convencionalmente chamado de "nível de desdobramento funcional da *feature*", que aqui são dois: Elementar e Parcial. *Features* do nível Elementar são as mais simples, aquelas que executam uma função "indivisível" no projeto. *Features* do nível Parcial, são aquelas que podem ser modeladas com duas ou mais *features* do nível Elementar, mas que são freqüentemente usadas em conjunto pelo projetista. A partir dos níveis Parciais e Elementares as *features* são agrupadas em conjuntos de definições conceitualmente similares para o projetista, tais como: "Furos", "Rebaixos", "Canais", etc. A estrutura de classificação finaliza com o desdobramento de cada grupo, em suas *features* constituintes.

Na Figura 6.4 isto é exemplificado nos ramos "Adicionais - Corte - Reto - Elementar e Parcial".

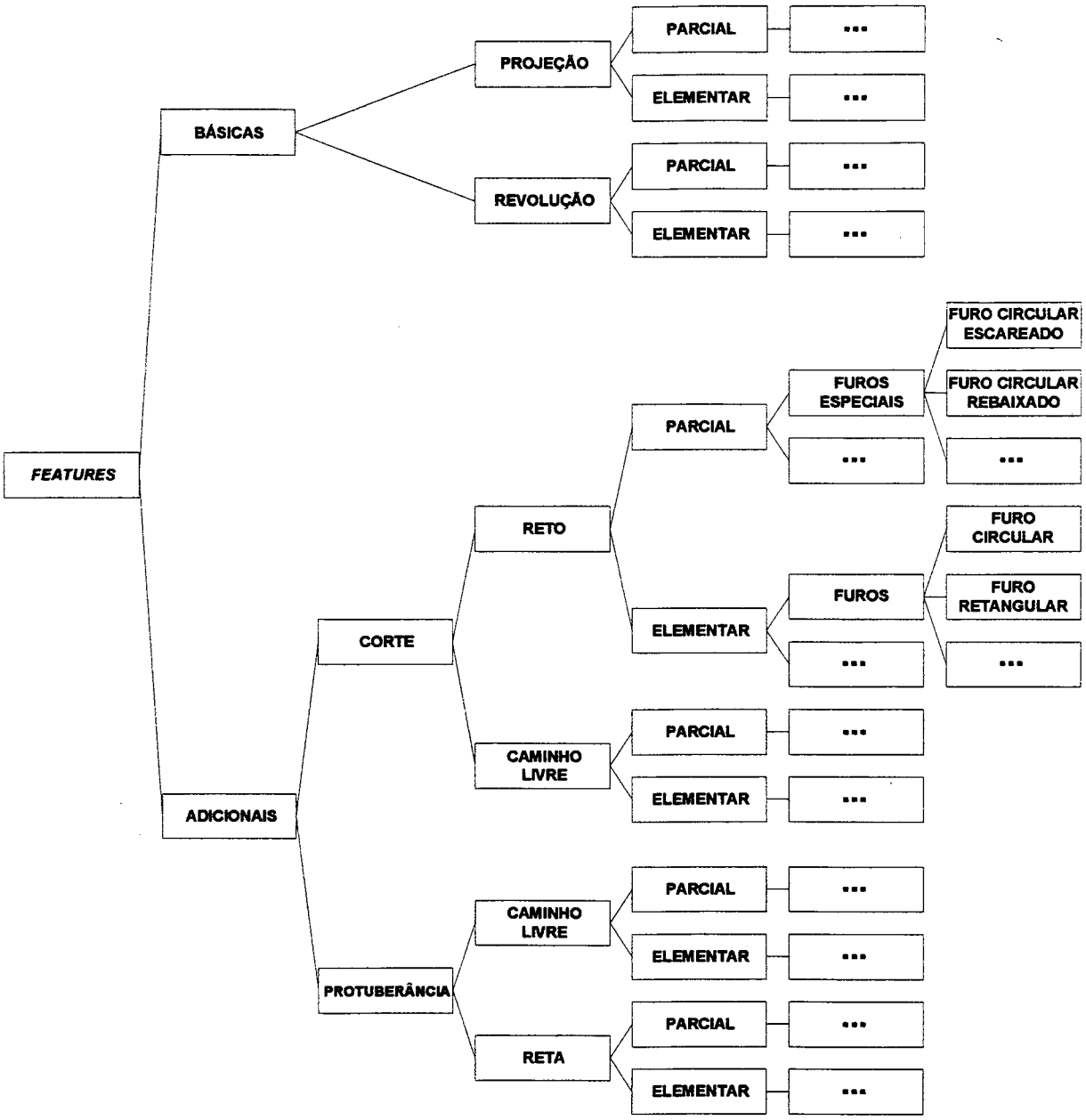
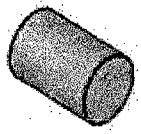
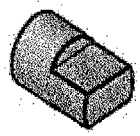
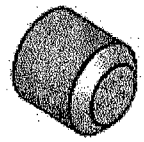


Figura 6.4: Classificação das *features* identificadas e formalizadas

O resultado é mostrado na Tabela 6.1. Na primeira coluna desta tabela está registrado a classificação da *feature* que é especificada através de um ícone e nome nas colunas dois e três. Os parâmetros dimensionais são registrados na coluna quatro que se subdivide em: Normalizados, Independentes e Dependentes. O significado desta divisão está de acordo com a classificação de variáveis proposta no item 4.3.3. Na quinta coluna estão registrados os tipos de atributos tecnológicos que serão associados á esta *feature*. A

biblioteca completa com todas as features identificadas e formalizadas é apresentada no Anexo 1 através de tabelas seguindo esta mesma formatação.

Tabela 6.1: *Features* do Eixo do Pinhão da Betoneira 120L.

CLASSIFICAÇÃO				FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
						Normalizados	Indep.	Dependentes	
BÁSICAS	PROJEÇÃO	BARRAS			Barra cilíndrica	Diâmetro	Comprimento	-	Peso (Kg/m) Material
ADICIONAIS	CORTE	RETO	CANAIS RETOS		Rasgo para chaveta deslizante	Raio da fresa Profundidade Comprimento	-	-	Processo Ferramenta
			REBAIXOS E ENTALHES		Espiga	-	Espessura Comprimento	Diâmetro	Processo Ferramenta
		CAMINHO LIVRE	ESCALONAMENTOS		Escalona - mento Cilíndrico	Raio Altura Largura	-	-	Processo Ferramenta
			CANAIS E RASGOS CIRCULARES		Canal retangular	Profundidade Largura	-	-	Processo Ferramenta
			CHANFRO QUINA CIRC.		Chanfro em quina circular	Lado 1 Lado 2	-	Comprimento Ângulo	Processo Ferramenta

É possível fazer algumas simulações de modelamento a partir de esboços das *features* usando caneta e papel. Segundo SHAH, J. J. e MÄNTYLÄ, M. (1995), isso é necessário para obter opiniões dos usuários, antes da implementação computacional da biblioteca. Contudo, como o objetivo deste trabalho é explorar ferramentas já existentes em sistemas CAD comerciais, estes aspectos são trabalhados durante o armazenamento das definições no sistema.

6.4.2 Arquivamento das definições no sistema CAD

Conforme descrito anteriormente, no MicroStation/J Modeler V7.1, a biblioteca de *features* é armazenada na forma de arquivos com perfis 2D parametrizados, que podem ser facilmente reusados através de sua associação com as *Features* Genéricas. Os perfis 2D das *features* identificadas e formalizadas foram parametrizados e armazenados. Todos eles foram usados em um modelamento experimental no ambiente CAD, adicionando restrições de posicionamento e verificando o comportamento do modelo quanto à manutenção da intenção de projeto através execução de algumas alterações em parâmetros dimensionais.

A Figura 6.5 ilustra a seqüência de modelamento do Eixo do Pinhão da Betoneira 120L. Nesta figura pode ser observada a associação entre perfis 2D e *Features* Genéricas para geração das formas geométricas das *features* personalizadas. A cada passo do processo de modelamento, é adicionada uma característica geométrica ao modelo, com uma função de projeto específica. No primeiro passo é definido o corpo da peça, no passo dois é adicionado um chanfro para quebra de quina, no terceiro passo são adicionados dois rebaixos na ponta do eixo para encaixe, e assim por diante [8].

Na construção da biblioteca, foi verificado que o número de perfis é sempre menor ou igual ao número de *features* que podem ser modeladas com eles, já que um mesmo perfil pode ser associado a diferentes *Features* Genéricas, e assim gerar diferentes formas geométricas.

Como contribuição, este estudo de caso permitiu o desenvolvimento de duas técnicas de parametrização com as ferramentas do MicroStation/J Modeler V7.1, que podem melhorar o modelamento da intenção de projeto em alguns casos específicos. Na explicação das técnicas que será dada a seguir, considera-se que o leitor já tenha um entendimento do processo de parametrização do sistema CAD em uso (MicroStation/J Modeler V7.1). Caso contrário, pode-se consultar os manuais disponibilizado gratuitamente no site da Bentley (http://docs.bentley.com).

A primeira técnica deve ser usada quando o posicionamento de uma *feature* é dependente de algum parâmetro dimensional de outra *feature*. Ela consiste em deslocar o ponto de referência para posicionamento do perfil 2D e transformar a distância de deslocamento deste ponto, em um parâmetro que possa ser referenciado. Esta técnica foi usada para modelar o perfil longitudinal do rasgo para chaveta deslizante ilustrado na Figura 3.10 (c). Neste caso, a posição da *feature* "Rasgo para chaveta deslizante" é dependente do diâmetro do eixo e seu perfil foi modelado conforme a Figura 6.6. Assim, a distância de

[8] As dimensões colocadas em cada perfil 2D na Figura 6.5, são meramente ilustrativas.

deslocamento do ponto de referência pode ser associado ao diâmetro do eixo utilizando as ferramentas para definição e referenciamento de variáveis globais.

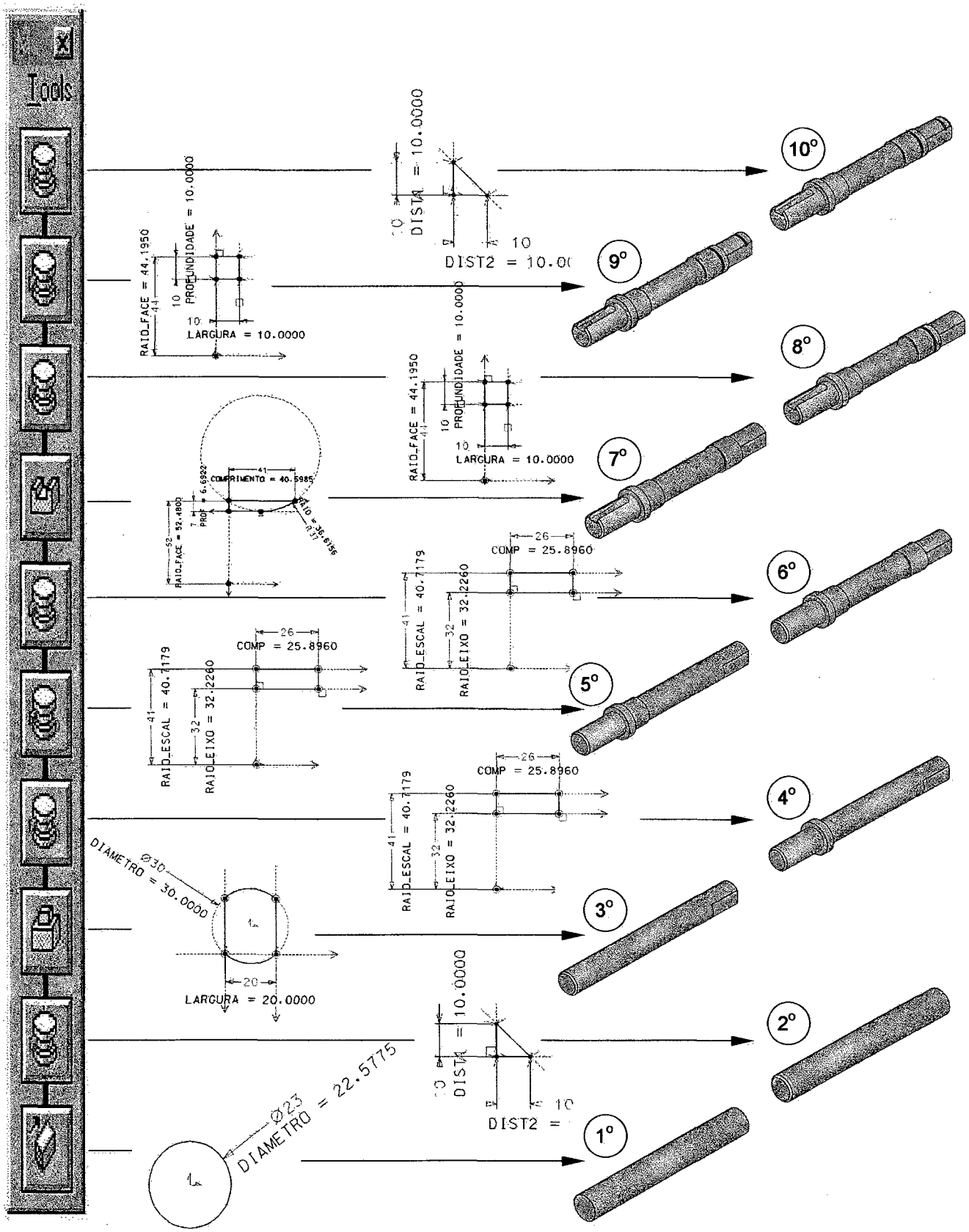


Figura 6.5: Modelamento do Eixo do Pinhão da Betoneira 120L

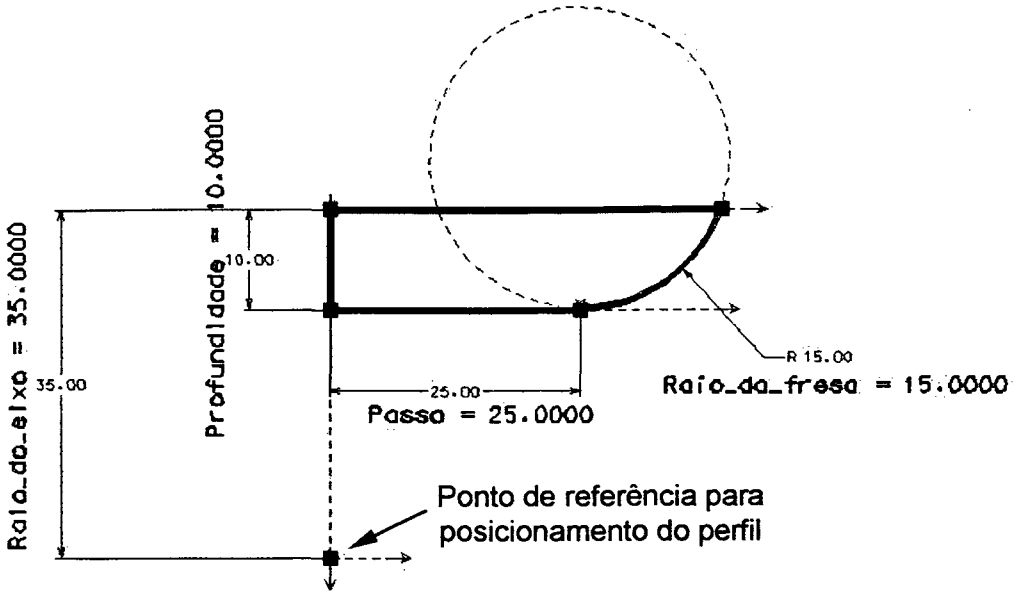


Figura 6.6: Deslocamento do ponto de referência para posicionamento do perfil 2D para auxiliar o modelamento da intenção de projeto

A segunda técnica deve ser usada quando existe um parâmetro no perfil 2D que, de acordo com a intenção de projeto, nunca deve ser alterado. Ela permite que tal parâmetro seja fixado através restrições geométricas, invisíveis para o usuário do perfil durante o modelamento. Para este caso a Figura 6.7 ilustra o perfil da ponta do eixo central da Betoneira 120L, cuja forma geométrica é mostrada no modelo 3D ao lado do perfil, em tamanho reduzido. Este perfil possui dois parâmetros fixos: base e chanfro. A técnica é explicada à seguir para o caso do chanfro.

- Uma vez que o perfil tenha sido normalmente parametrizado são traçadas duas linhas paralelas de construção (linhas tracejadas), conforme ilustra o quadro ampliado da Figura 6.7. A primeira linha deve passar pelo ponto de âncora do perfil (ponto 1), e um ponto do elemento geométrico que define o parâmetro a ser fixado, ponto 2. A segunda linha de construção é traçada passando pelo segundo ponto do elemento geométrico a ser fixado, ponto 3.
- O segundo passo consiste em fixar a distância entre as duas linhas de construção paralelas. Isso é feito fixando um ponto na extremidade da segunda linha previamente traçada. No quadro ampliado da Figura 6.7, é o ponto 4.
- O terceiro passo consiste em estabelecer uma restrição geométrica que faça com que o ponto inicial e final do elemento geométrico que define o parâmetro a ser fixado, sempre estejam em contato com as linhas paralelas que os tangenciam. Estas restrições devem ser adicionadas em todos os pontos (1, 2, 3 e 4).

O mesmo procedimento é usado para fixar a dimensão da base da ponta do Eixo Central da Betoneira 120L. Contudo, note que para este parâmetro, uma das linhas de construção paralelas coincide com uma linha do perfil, por isso não aparece tracejada.

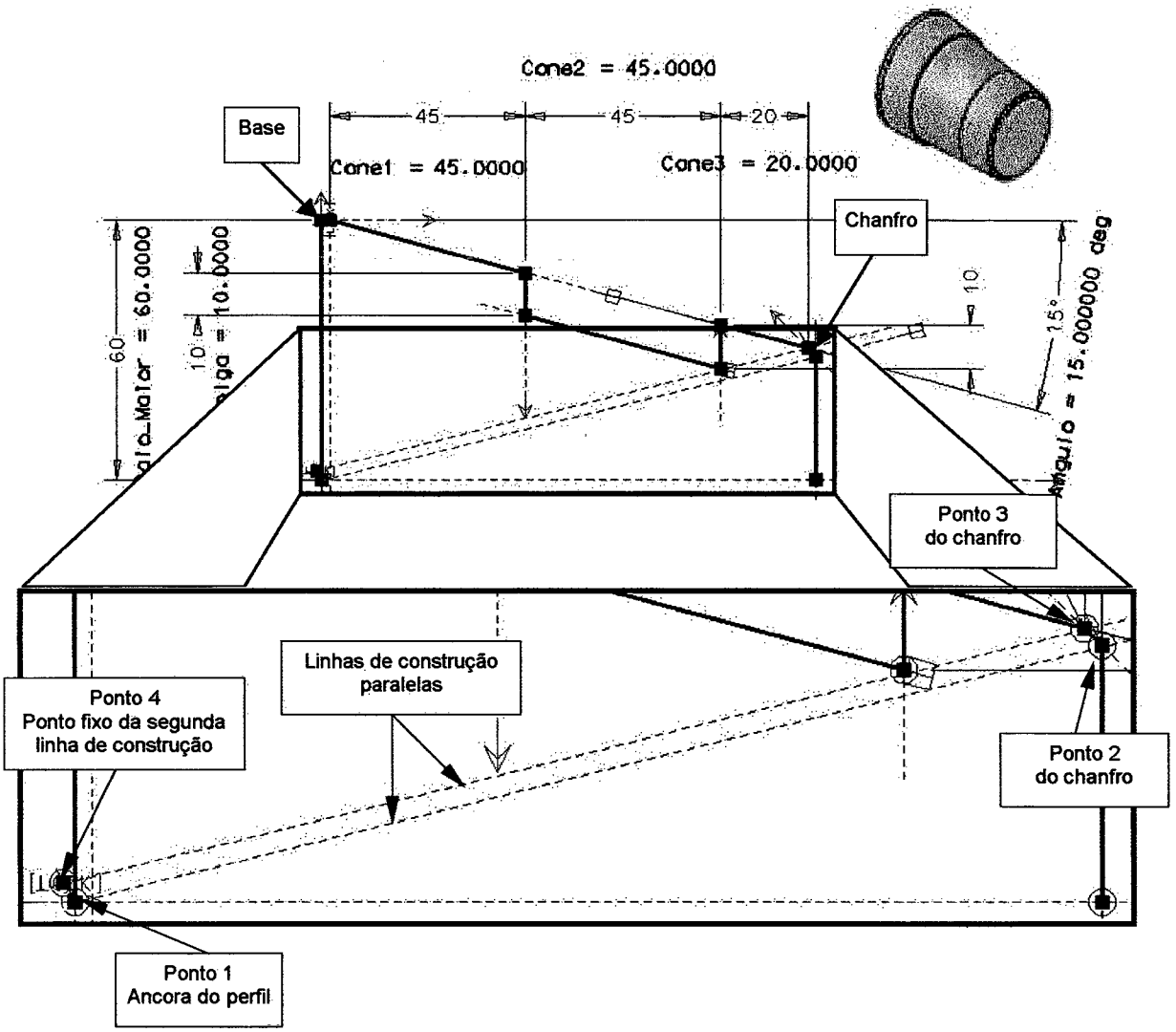


Figura 6.7: Fixação de parâmetros por meio de restrições geométricas

6.4.3 Registro das informações não geométricas no RDBMS

As informações não geométricas consistem de parâmetros dimensionais padronizados e atributos tecnológicos das *features*. Alguns destes dados foram anotados durante o processo de identificação e formalização das mesmas. Eles podem ser adquiridos por meio de entrevistas com os projetistas da empresa, consulta a catálogos de fornecedores, normas internas ou externas. Seu registro no RDBMS significa a geração dos grupos de tabelas que compõem o "BD reusáveis", cujo esquema de organização foi apresentado no item 5.3.1.

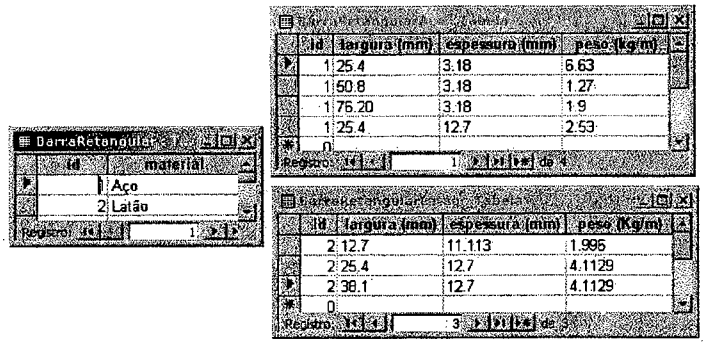


Figura 6.8: Tabelas com dados não geométricos no "BD reusáveis"

A Figura 6.8 ilustra algumas tabelas com informações não geométricas da Feature Básica "Barra Retangular" adquiridas no Projetista de Máquinas (PROVENZA, 1985).

6.5 IMPLEMENTAÇÃO DA INTERFACE CAD-RDBMS

A Figura 6.9 mostra a janela principal da Interface CAD-RDBMS. Esta janela possui menus com algumas ferramentas básicas e botões que ativam o processo de construção de cada *feature*. Tais botões são estruturados em painéis que refletem a classificação da *feature* correspondente, conforme estrutura descrita anteriormente.

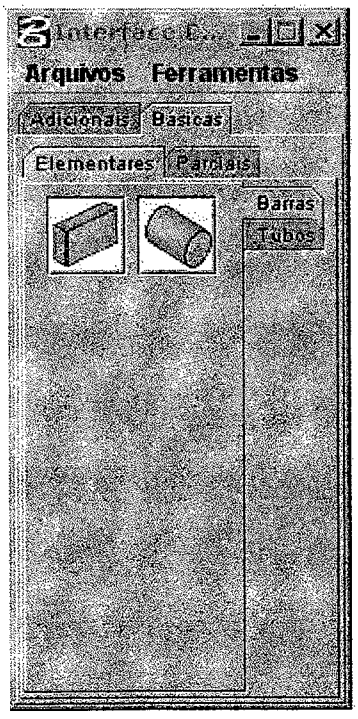


Figura 6.9: Janela principal da Interface CAD-RDBMS

Na Figura 6.10, o usuário iniciou o modelamento de uma peça com a *feature* Básica Barra retangular, ativando sua caixa de diálogo com um clique em seu botão na janela principal. A caixa de diálogo possui campos de texto, para especificar o valor das variáveis de definição da *feature* e recursos para o acesso e uso dos dados não geométricos, armazenados no banco de dados "BD reusáveis". As caixas de diálogo de todas as *features* seguem este mesmo formato padrão.

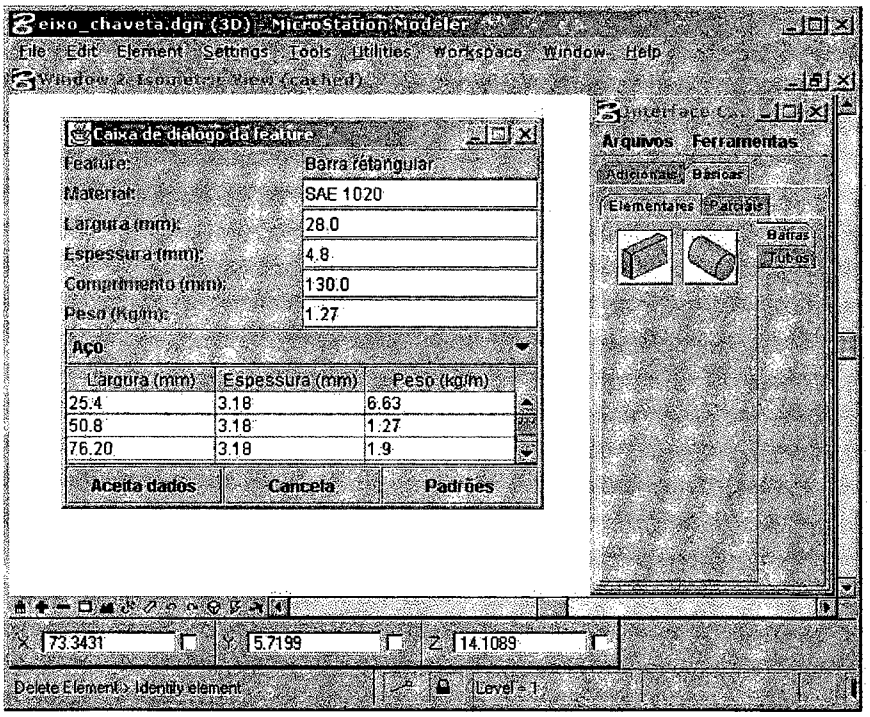


Figura 6.10: Caixa de diálogo da *feature*

Tendo sido declaradas todas as variáveis necessárias para a definição da *feature*, o usuário as aceita e seus parâmetros dimensionais são enviados para funções que controlam as ferramentas para modelamento de *features* no sistema CAD. Então, o perfil da seção transversal ou longitudinal da *feature* em questão é exibido para posicionamento, conforme mostrado na Figura 6.11.

Definido o posicionamento, a forma geométrica da *feature* é automaticamente modelada. A Figura 6.12 ilustra um suporte da betoneira 120L modelado através da Interface CAD-RDBMS. Na janela principal estão evidenciados algumas *features* do grupo "Furos".

Em qualquer momento do processo de modelamento é possível visualizar os dados não geométricos do modelo conforme mostrado na Figura 6.13. Também existem algumas ferramentas para a edição destes dados. Tais ferramentas permitem salvar a

árvore de informações não geométricas em um arquivo específico, eliminar algumas *features* desta estrutura de dados e adicionar outras, se necessário.

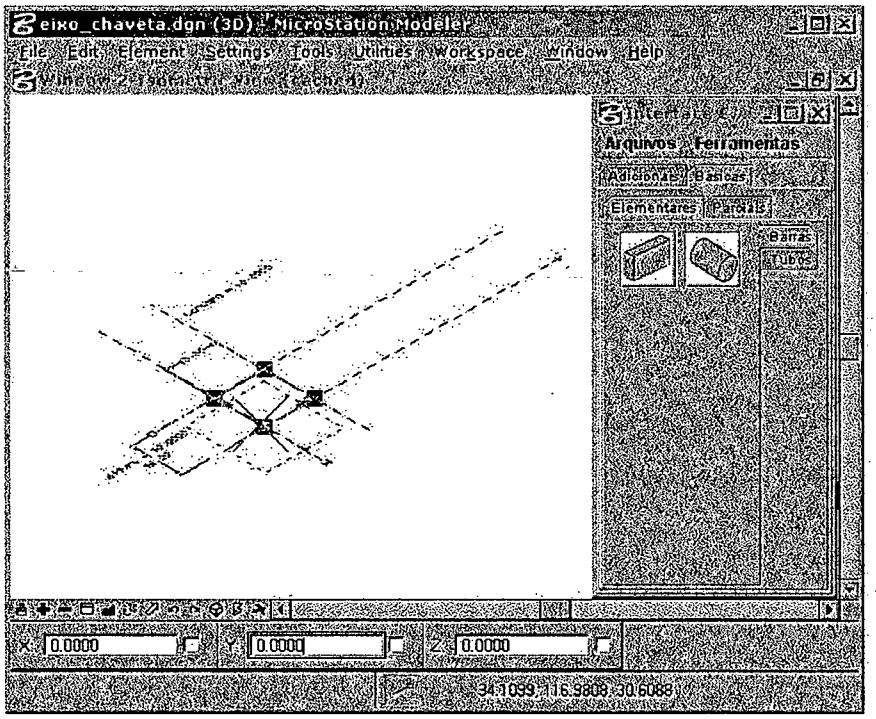


Figura 6.11: Posicionamento da *feature*

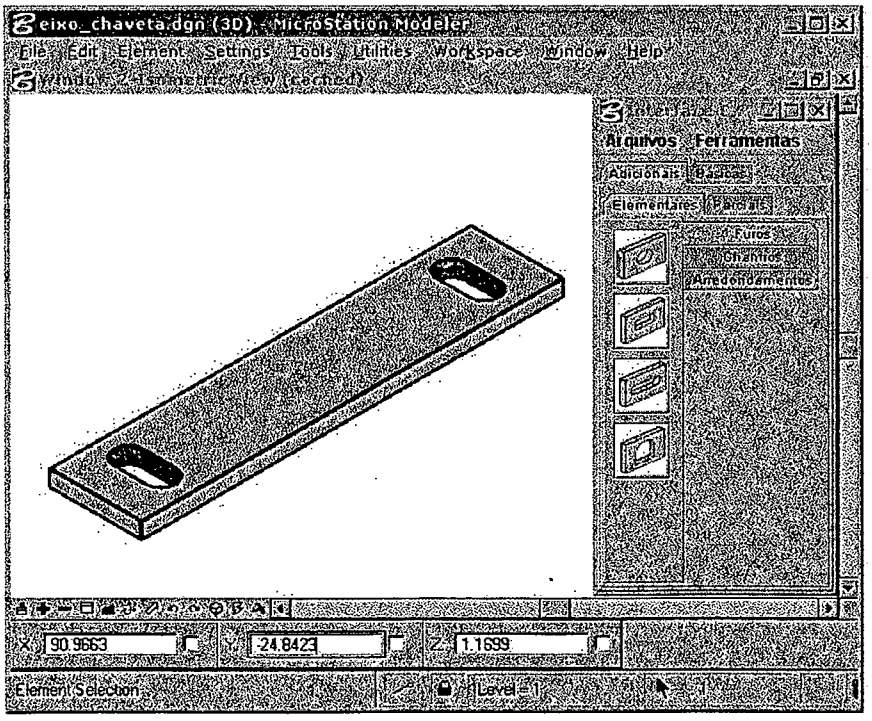


Figura 6.12: Peça modelada através da Interface CAD-RDBMS

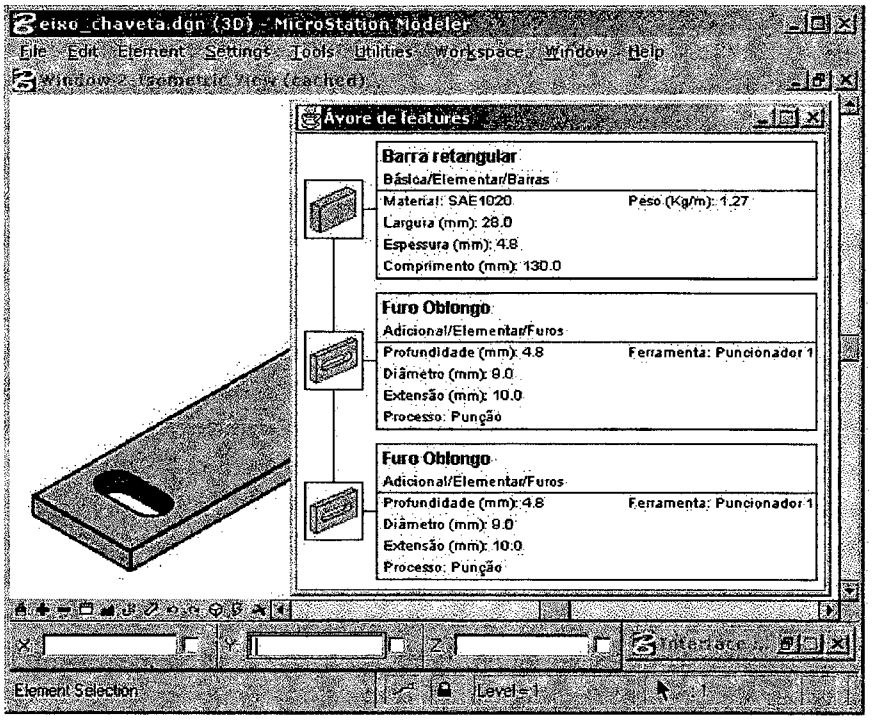


Figura 6.13: Visualização dos dados não geométricos da peça modelada

Estes dados podem ser publicados no: "BD reusados", após a finalização do processo de modelamento. A Figura 6.14 ilustra os registros deste banco de dados para a peça modelada.

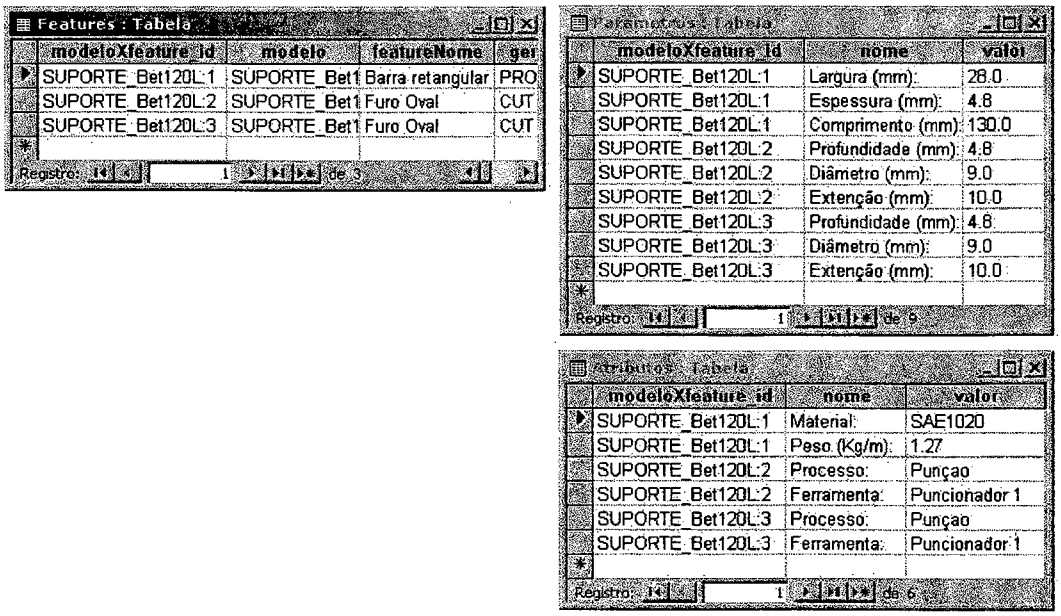


Figura 6.14: Registro dos dados não geométricos no "BD reusados"

6.5.1 Detalhes da implementação

A Figura 6.15 ilustra com detalhes, os componentes da Interface CAD-RDBMS. Eles são organizados em dois módulos. O "Módulo 1" utiliza a linguagem JMDL, e implementa as funções de interação com o usuário e RDBMS. O "Módulo 2" é escrito em MDL, ele implementa as funções de manipulação das ferramentas para modelamento de *features* presentes no MicroStation/J Modeler, as quais agem diretamente na base da dados nativa do sistema CAD cujo esquema é denominado *MicroStation Design File* (DGN).

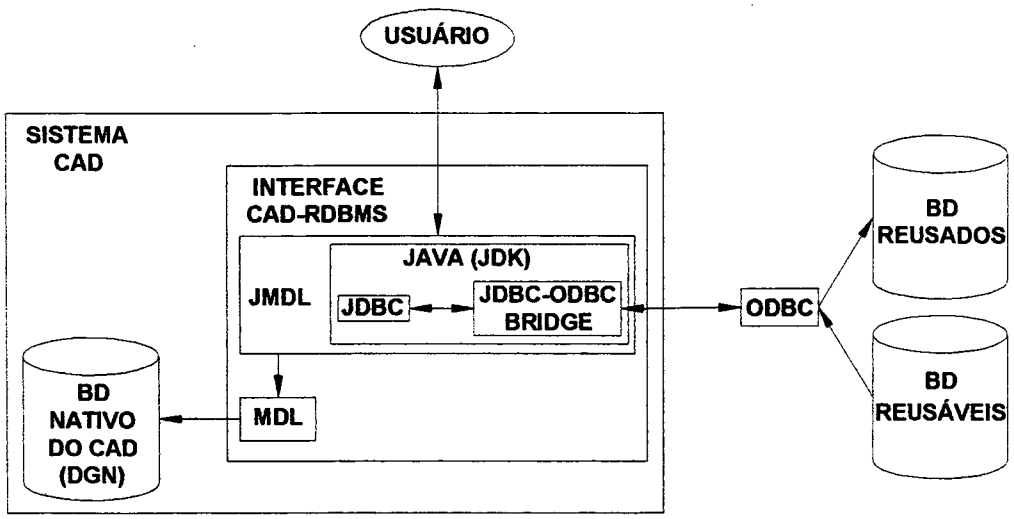


Figura 6.15: Componentes e fluxo de informação na Interface CAD-RDBMS

No Módulo 1, para a implementação da interface de interação com o usuário foram utilizadas as classes do pacote "javax.swing", pertencente ao JDK (*Java Development Kit*). As funções de interação com o RDBMS e associação dos dados geométricos com os não geométricos, são implementadas com recursos do pacote "java.sql", também pertencente ao JDK. As classes deste pacote interagem com as interfaces JDBC e JDBC-ODBC Bridge, usadas para conexão com banco de dados relacionais que aceitam o padrão de comunicação ODBC. Maiores informações sobre estes pacotes podem ser encontradas nas referências (HORSTMANN e CORNELL, 1999), (HORSTMANN e CORNELL, 2000).

As funções de transferência dos dados não geométricos para o sistema CAD são implementadas de forma específica para cada *feature*. Para tal elas são encapsuladas em uma hierarquia de classes modeladas com técnicas de programação orientada ao objeto.

Uma classe abstrata denominada "ObjectFeature" implementa comportamentos comuns a todas as *features* e define três métodos abstratos que são particularmente importantes para o entendimento da Interface CAD-RDBMS, são eles:

- "void setVariáveis(String[] s)": método que recebe um vetor de "String" com todos os parâmetros para definição da *feature* e os usa para inicializar suas variáveis.

- "Variavel[] getVariáveis()": método que retorna um vetor de "Variavel" com os parâmetros das variáveis da classe. O tipo de dado ou classe: "Variavel", foi implementado para tornar possível um tratamento padronizado para variáveis dependentes, normalizadas e independentes de cada *feature*.
- "String getProtocol()": método que retorna um protocolo usado para comunicação com as funções que manipulam as ferramentas de modelamento de *features* no sistema CAD.

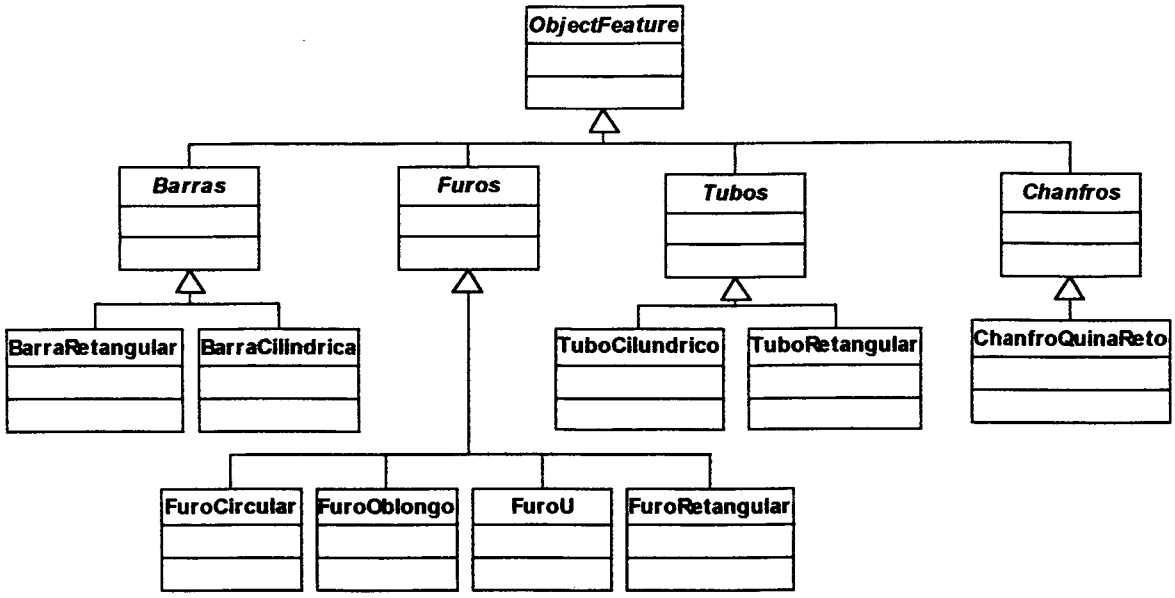


Figura 6.16: Relacionamento hierárquico das classe que encapsulam os comportamentos das *features* na Interface CAD-RDBMS

Da classe ObjectFeature derivam outras, também abstratas, que implementam os comportamentos comuns a grupos de *features* conceitualmente similares para o projetista, denominadas: "classes grupo". Seguindo a mesma abordagem usada para classificação das *features*, cada "calsse grupo" é denominada com um nome genérico, tal como: "Furos", "Rebaixos", "Barras", etc. De cada uma delas derivam outras, instanciáveis, que implementam métodos abstratos definidos nas classes pai e outros métodos locais, sempre de acordo com as características específicas de uma única *feature* personalizada. Tais classes são denominadas: "classes *feature*". Este relacionamento hierárquico é ilustrado na Figura 6.16 utilizando o padrão UML (*Unifield Modeling Language*). São mostradas apenas as classes implementadas para a experimentação da proposta no presente trabalho.

No "Módulo 2", as funções que manipulam as ferramentas de modelamento de *features* no sistema CAD são escritas em MDL. Para tal, foram utilizados, além do API padrão, um outro API específico para a configuração Modeler do MicroStation/J V7.1

chamado ModAPI, fornecido pela Bentley como exemplo de programação MDL. Neste módulo são implementadas as funções necessárias para manipulação dinâmica dos perfis 2D, aquisição dos dados advindos da interface e criação das *Features* Genéricas: Projeção, Corte reto e Protusão reta.

As funções de aquisição dos dados da interface têm como parâmetro de entrada um vetor de caracteres, que é equivalente á uma "String" na linguagem JMDL. Neste vetor, são representados através de um protocolo de comunicação específico, descrito no Anexo 2, o nome da *feature* genérica, nome do perfil, parâmetros construtivos e dimensionais da *feature*.

Através de uma função especial chamada "mdl command", já disponibilizada pelo API padrão MDL, é possível passar parâmetros de entrada para funções específicas do ambiente MicroStation, na forma de comandos de linha ("keyins"), obedecendo a seguinte sintaxe: "mdl command <nome da função> <parâmetros de entrada>". O JMDL por sua vez, possui uma função denominada: "sendKeyin(String s)", que lança um "keyin" no MicroStation/J. Portanto, é possível usar o método "sendKeyin(String s)" para enviar a String retornada pelo método "String getProtocol()" de cada classe *feature*, para o método de aquisição dos dados advindos da interface. Assim se estabelece um meio de comunicação entre os dois módulos constituintes da Interface CAD-RDBMS.

O código fonte do "Módulo 2" está impresso no Anexo 3. Os códigos fonte e compilados de toda a Interface CAD-RDBMS, os banco de dados "DB reusáveis" e "BD reusados", mais os perfis 2D parametrizados, estão no CD anexado à este trabalho.

Os componentes principais da Interface estão organizados em uma estrutura de diretórios idêntica à mostrada na Figura 6.17. No diretório principal: InterfaceCADRDBMS, estão os arquivos com os banco de dados e os perfis 2D parametrizados. O "Módulo 1" do aplicativo está, logicamente, no diretório "Modulo1". Este diretório contém a classe principal do aplicativo denominada InterfaceCADRDBMS.mjava e outros 4 sub-diretórios: "access", "features", "gui" e "system". No "access", está a classe que implementa os métodos de interação com o RDBMS MS Acsess 97. No "features", estão todas as classes: "ObjectFeature", "classes grupo", "classes *features*" e a classe "Variável", mencionada anteriormente. No sub-diretório "gui", estão as classes que criam a interface gráfica de interação com o usuário. Ele possui dois outros sub-diretórios: "featureicons", com os arquivos dos ícones das features implementadas e "models", com modelos personalizados de componentes da interface. O diretório "system", possui apenas uma classe usada para implementar métodos usados por classes que estão em diferentes diretórios. O diretório: "Modulo2" possui os arquivos com o código fonte MDL e também o código compilado.

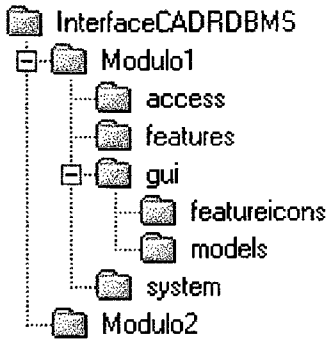


Figura 6.17: Estrutura de diretórios da Interface CAD-RDBMS

Para instalação do aplicativo é necessário ter em mãos o sistema CAD MicroStation/J Modeler V7.1. Caso o API ModAPI não tenha sido compilado, é preciso compilá-lo antes. Para isso é necessário alguns ajustes no Autoexec.bat. Tais ajustes são muito bem explicados nos manuais de programação MDL, gratuitamente disponibilizados pela Bentley. Para facilitar, no CD anexado existe um arquivo chamado Ajustes.txt, que possui as linhas de comando que devem ser adicionadas ao Autoexec.bat. Também é necessário registrar o *drive* ODBC para cada banco de dados. Este procedimento é realizado utilizando o aplicativo "ODBC (32bit)", que está disponível no Painel de Controle do Windows 98.

Para executar o aplicativo, o usuário deverá executar a classe InterfaceCADRDBMS.mclass do "Modulo 1", ler a aplicação MDL do Módulo 2 e anexar a biblioteca de células no ambiente do MicroStation/J Modeler V7.1.

```
...
private void composeTabbedPaneFeature()
{  //Adição das features
    tpf.addFeature(new BarraRetangular());
    tpf.addFeature(new BarraCilindrica());
    tpf.addFeature(new TuboRetangular());
    tpf.addFeature(new TuboCilindrico());
    tpf.addFeature(new FuroCircular());
    tpf.addFeature(new Chanfro());
    tpf.addFeature(new Arredondamento());
    tpf.addFeature(new FuroRetangular());
    tpf.addFeature(new FuroOval());
    tpf.addFeature(new FuroU());
   //***
}
...
```

Figura 6.18: Trecho do código fonte onde são adicionadas as *features* do aplicativo

Para a adição de novas *features* é necessário implementar as classes correspondentes obedecendo o modelo de herança mostrado anteriormente e alguns

padrões de variáveis, que podem ser facilmente verificados observando o código fonte (comentado) disponível no CD. Uma vez implementadas, tais classe devem ser armazenadas no diretório "*features*". Elas são adicionadas na interface chamando seu método construtor dentro do método "composeTabbedPaneFeature(), pertencente à classe "InterfaceCADRDBMS". A Figura 6.18, mostra o código fonte deste método, onde as *features* implementadas neste trabalho são adicionadas ao aplicativo: Interface CAD-RDBMS.

6.6 CONCLUSÕES

Este capítulo apresentou a implementação prática da abordagem de integração CAD-RDBMS descrita no capítulo anterior. Foi demonstrado na prática a viabilidade de se criar bibliotecas de *features* personalizadas com base em recursos genéricos já disponíveis em sistemas CAD comerciais de médio e grande porte. As características inerentes à tecnologia de *features* permite a criação de modelos com informações especializadas, o uso de um banco de dados externo para o armazenamento destas informações é uma abordagem viável.

O aplicativo Interface CAD-RDBMS acelera o processo de construção do modelo. Isso ocorre não só por fornecer uma interface intuitiva com acesso à parâmetros construtivos padronizados, mas também e principalmente, por automatizar o uso simultâneo das ferramentas genéricas para o modelamento de *features*, através do encapsulamento do processo construtivo de cada uma delas em uma única *feature* personalizada.

Os resultados apresentados neste capítulo representam a concretização dos objetivos inicialmente propostos para este trabalho. Portanto, eles merecem ser discutidos de forma mais abrangente, considerando os vários aspectos abordados no trabalho como um todo. O próximo capítulo apresenta essa discussão.

7 RESULTADOS E CONSIDERAÇÕES FINAIS

7.1 INTRODUÇÃO

Este trabalho apresentou um estudo sobre características dos sistemas CAD comerciais, aplicando a tecnologia de *features* em associação com RDBMS em uma nova abordagem para modelamento de peças no projeto mecânico. Foi proposto um aplicativo computacional, denominado Interface CAD-RDBMS, que combina recursos disponíveis em sistemas CAD de médio ou grande porte com RDBMS num ambiente de modelamento baseado em bibliotecas de *features* personalizadas. As *features* personalizadas encapsulam informações geométricas e não geométricas de projeto. Os esquemas de representação das informações no ambiente CAD e RDBMS e algumas características do modo de comunicação entre eles foram descritos em termos genéricos. Também foi apresentado a seqüência de atividades necessárias para implementação do aplicativo.

A proposta foi experimentada, com a implementação das funcionalidades básicas da Interface CAD-RDBMS. A biblioteca de *features* personalizadas foi criada a partir de desenhos de projeto de uma Betoneira de 120L, fornecidos pela Metalúrgica Erwino Menegotti Ltda., localizada em Jaraguá do Sul - SC. Utilizou-se o sistema CAD MicroStation/J Modeler V7.1 da Bentley e o RDBMS Access 97 da Microsoft.

Nos próximos itens estes resultados são analisados e discutidos, buscando ressaltar suas conseqüências e contribuições. Por fim é feita uma descrição de propostas para estudos futuros que possam se referenciar ao presente trabalho.

7.2 ANÁLISE DOS RESULTADOS

O estudo realizado para o desenvolvimento deste trabalho permitiu visualizar as potencialidades e limitações na utilização dos recursos disponíveis em sistemas CAD comerciais para a implementação de aplicações baseadas na tecnologia de *features*. Sistemas CAD comerciais de médio e grande porte oferecem uma abordagem para modelamento geométrico de sólidos com ferramentas baseadas na tecnologia de *features*. Tais ferramentas permitem a criação de bibliotecas de definições personalizadas com recursos genéricos de modelamento. É possível representar a intenção de projeto, no que diz respeito à forma geométrica, utilizando ferramentas para estabelecimento de

relacionamentos ou restrições entre as entidades geométricas que definem uma ou mais *features*. Também é possível associar dados não geométricos de projeto ao modelo geométrico, utilizando EEDs ou base de dados externas.

Estes recursos possibilitam a criação de modelos no sistema CAD, associados a informações geométricas e/ou não geométricas específicas que permitem o desenvolvimento de aplicações especializadas. Contudo, elas são implementadas de um modo fragmentado, ou seja, as definições de *features* não encapsulam estas informações na forma de uma única entidade de modelamento. Deste modo, a base de dados do sistema CAD continua centrada em entidades geométricas de baixo nível semântico tais como: pontos, linhas e curvas. As definições de *features* estão presente apenas na interface de interação com o usuário.

Esta "deficiência" deve ser superada nas novas gerações de sistemas CAD, com a criação de bases de dados centradas em entidades de alto nível semântico. Por agora, é possível e viável desenvolver aplicações especializadas e úteis para as atividades de projeto, usando as ferramentas já disponibilizadas pelos desenvolvedores do sistema CAD. A abordagem de exploração da tecnologia de *features* presente em sistemas CAD comerciais proposta neste trabalho, adota esta estratégia.

A abordagem propicia a implementação de um aplicativo computacional de auxílio ao projeto. Tal aplicativo é capaz de melhorar os modelos de produtos normalmente criados por sistemas CAD comerciais, pela associação dos mesmos a uma base de dados relacional, contendo parâmetros dimensionais e atributos tecnológicos do produto modelado. Estes atributos tecnológicos podem ser de vários tipos e associados a diferentes campos de aplicação da engenharia, tais como: planejamento do processo de fabricação, análise de custos, análise de resistência, etc.

O aplicativo computacional pode interagir com dados reusáveis de projeto armazenados em um banco de dados relacional, denominado: "BD reusáveis", para auxiliar a construção dos modelos do produto. Este banco de dados, pode ser alimentado por pessoas que trabalham em diferentes áreas de conhecimento, atuantes no processo de desenvolvimento do produto. Além disso, a base de dados não geométricos associada aos modelos gerados, denominada "BD reusados", pode ser facilmente acessada por outras aplicações. Deste modo, cria-se um fluxo de informações entre várias áreas do sistema de manufatura. Portanto, a abordagem proposta neste trabalho é uma alternativa para o desenvolvimento de um ambiente computacional de suporte à Engenharia Simultânea.

Os esquemas de representação das informações de projeto fazem uso de recursos comumente disponíveis em sistemas CAD comerciais de médio ou grande porte e funcionalidades básicas de RDBMS. Deste modo, sua implementação é independente de um sistema computacional específico.

Tais esquemas de representação oferecem um modo para definição de *features* personalizadas em termos de ferramentas genéricas preestabelecidas. Assim, apesar das informações serem manipuladas pelo usuário, na forma de entidades altamente personalizadas, não há alteração da estrutura da base de dados nativa do sistema CAD, nem é adicionada nenhuma dificuldade extra para a troca ou compartilhamento dos dados geométricos modelados. Por outro lado, o compartilhamento destes dados entre diferentes sistemas CAD poderia ser facilitado se os desenvolvedores de cada sistema adotassem um mesmo padrão de entidades genéricas para representação de *features* personalizadas.

O fato de ser usado apenas ferramentas preestabelecidas nos sistemas computacionais acarreta na possibilidade de se criar manualmente modelos idênticos aos criados através da Interface CAD-RDBMS. Na verdade, os estudos apresentados neste trabalho também servem como referência para o entendimento do modo de utilização da tecnologia de *features* presente em sistemas CAD comerciais. Contudo, uma grande vantagem da implementação prática desta proposta é que ela aumenta a velocidade de construção dos modelos por constituir-se de uma interface intuitiva, com recursos que possibilitam o reuso de várias ferramentas de maneira unificada e automatizada.

Apesar do caráter experimental do aplicativo Interface CAD-RDBMS, ela é foi capaz de validar a abordagem proposta. Contudo, para sua aplicação comerciai será necessário um maior desenvolvimento. Faltam implementar os métodos para manipulação automática das *features* Genéricas: Corte e Protutuberância de caminho livre, ferramentas de estabelecimento de relacionamento geométricos e variáveis globais. Também é necessário otimizar os algoritmos uma vez que a linguagem JMDL demostrou-se lenta e instável.

7.3 DIFICULDADES ENCONTRADAS

Para discussão das dificuldades encontradas no desenvolvimento deste trabalho é necessário destacar seu caráter investigativo. Trabalhos referentes à exploração da tecnologia de *features* já existente em sistemas CAD comerciais não são facilmente encontrados na literatura. Foi necessário um estudo aprofundado sobre os recursos disponíveis em sistemas CAD comerciais e a presença da tecnologia de *features* neste âmbito. Discernir as generalidades das especificidades conceituais foi a primeira dificuldade encontrada. Entender como tais conceitos têm sido empregados comercialmente, foi outro desafio. Espera-se que estes aspectos tenham sido satisfatoriamente resolvidos nos Capítulos 2 e 3, de revisão bibliográfica.

Outra dificuldade encontrada diz respeito às linguagens JMDL e MDL usadas para implementação da Interface CAD-RDBMS. A documentação disponibilizada pela Bentley sobre a linguagem JMDL não é muito detalhada. Além disso, no atual estágio de desenvolvimento, sua biblioteca de classes não lida com entidades do módulo Modeler do MicroStation/J. O MDL possui funções para o Modeler, contudo a documentação destas funções é inexistente. Além disso, faltam livros destinados ao ensino de programadores iniciantes nestas duas linguagens. Este tipo de dificuldade tem sido declarada por vários outros desenvolvedores do MicroStation, conforme descrito em artigo por SAHAI, R. S. (2000).

7.4 SUGESTÕES PARA TRABALHOS FUTUROS

Através da realização deste trabalho, foi possível identificar vários temas que necessitam maiores estudos. A seguir descreve-se resumidamente alguns deles como sugestões para trabalhos futuros:

- Aperfeiçoamento da Interface CAD-RDBMS. Sugere-se o uso da linguagem MDL para a implementação das interfaces de interação com o usuário, empregando a JMDL apenas na implementação da hierarquia de classes que encapsulam os comportamentos específicos de cada *feature* personalizada.
- Desenvolvimento de um banco de informações não geométricas reusáveis de projeto, estruturado em termos de *features*, para aplicações específicas. Em trabalhos futuros podem ser estudadas necessidades reais de algumas aplicações de projeto específicas, para construção de um banco de dados não geométricos mais completo.
- Desenvolvimento de aplicações que utilizem uma base de informações não geométricas de projeto registradas em um RDBMS e estruturadas em termos de *features* personalizadas para auxílio a atividades presentes no sistema de manufatura. Tais aplicações podem diminuir o tempo de desenvolvimento de um produto.
- Troca de dados de *features* personalizadas. A troca de modelos de features personalizadas entre sistemas CAD é um grande desafio. Conforme já comentado, a representação de features em termos de entidades genéricas, conforme proposta deste trabalho, parece ser uma abordagem promissora para resolução deste problema.
- Adição de semântica ao modelo geométrico do sistema CAD. Nem todos os sistemas CAD oferecem a possibilidade de programação com linguagens orientadas ao objeto. Em trabalhos futuros, pode ser desenvolvido uma ferramenta que permita a associação bidirecional de classes programadas em Java ou qualquer outra linguagem

de programação orientada ao objeto, à parâmetros construtivos das entidades geométricas de um sistema CAD qualquer. Isso possibilitará adicionar comportamentos inteligentes e personalizados ao modelo gerado no sistema CAD, sendo portanto, um poderoso recurso para modelagem de *features* personalizadas.

- Programação genérica em sistemas CAD - A implementação comercial de aplicações personalizadas, integradas a sistemas CAD, depende de um conhecimento especializado a respeito das linguagens de programação oferecidas pelo sistema. Isto acaba encarecendo estes tipos de desenvolvimentos. Existe a necessidade de estudos sobre abordagens de programação que otimizem a reusabilidade de códigos, quando um mesmo aplicativo for implementado em sistemas CAD de diferentes desenvolvedores.

8 REFERÊNCIAS BIBLIOGRÁFICAS

AN, D., LEEP, H. R., PARSEI, H. R. e NYALUKE, A. P., A product data exchange integration structure using PDES/STEP for automated manufacturing applications, **Computers Industrial Engineering**, Vol. 29, n. 1-4, pg. 711-715, 1995.

ASU Design Automation Lab, **http://ASUdesign.eas.asu.edu/**, 2000.

BACK, N. e FORCELLINI, F. A., **Projeto de Produtos**, Curso de Pós-Graduação em Engenharia Mecânica UFSC, 1999.

BHAT, S. K. e BEAGAN, D. D., Feature-based data management, **Mechanical Engineering**, March, 1989.

BIDARRA, R. e BRONSVOORT, W. F., Semantic feature Modelling, **Computer-Aided Design**, Vol. 32, pg. 201-225, 2000.

BRONSVOORT, W. F. e JANSEN, F. W., Feture modelling and conversion - Key concepts to concurrent engineering, **Computers in Industry**, Vol. 21, pg 61-86, 1993.

BUTTERFIELD, W., GREEN, M., SCOTT, D. e STOCKER, W., **Part Features for process planing**, Technical Report R-85-ppp-03, CAM-I, Inc., Arlington, Texas, 1985

CHANG, T-. C.; WYSK, R. A.; WANG, H-. P.; **Computer Aided Manufacturing**; 2° Edição; Prentice Hall; New Jersey; 1998.

CLARK, K. B e FUJIMOTO, T. **Product Development Performance: estrategy, organization and manangement in the world auto industry**. Bostom-Mass: Harvard Business Scholl Press, 1991.

CUNHA, R. R. M., **Estudo e desenvolvimento de metodologias na troca de dados em CAD/CAM**, Estudo dirigido, Departamento de Engenharia Mecânica, Universidade Federal de Santa Catarina, 2000.

CUNHA, R. R. M. e DIAS, A., Um esboço de uma nova sistemática de suporte às fases de processo de projeto aplicando sistemas CAD, **Anais do COCIM 2000**, 2000.

DANTE, C. J., **Introdução à Sistemas de Banco de Dados**, 4° Edição, Editora Campus, Rio de Janeiro – RJ, 1991.

DATAQUEST, End-User Analysis: Mechanical CAD/CAM/CAE From the User Viewpoint, 1999.

DONG, A. e AGOGINO, A. M., Mananging design information in enterprise-wide CAD using 'smart drawings', **Computer-Aided Design**, Vol. 30, No. 6, pg. 425-435, 1998.

DUAN, W.; ZHOU, J. e LAI, K.; FSMT: a feature solid-modelling tool for feature-based design and manufacture; **Computer-Aided Design**, Vol. 25, no. 1, pg. 29-38, 1993.

DURAN, O., ORSATO, A. e LEONHARDT, J., CADFEAT - sistema baseado em *features* para estruturas, **Anais do 8° Congresso Chileno de Engenharia Mecânica**, pg. 91-95, 1998.

EYNARD, B.; BELLOY, P. e LAFON, P., Towards an integration of knowledge in mechanical engineering, **Anais do IDMME**, 2000.

FERREIRA, J. C. E., BUTZKE, A. U., NETO, F. F., Um sistema CA para modelagem de uma família de peças em máquinas agrícolas, **Anais do VII Congresso nacional de Engenharia Mecânica - Chile**, pg. 101-104, 1997.

FENG, C., HUANG, C., KUSIAK, A. e LI, P., Representation of function and features in detail design, **Computer-Aided Design**, Vol. 28, no. 12, pg. 961-971, 1996

HETEM, V., Communication: Computer aided engineering in the next millennium, **Computer-Aided Design**, Vol. 32, pg. 389-394, 2000.

HORSTMANN, C. S. e CORNELL, G., **Core Java 2, Volume I – Fundamentals**, Sun Microsystems Press; Prentice Hall, 1999.

HORSTMANN, C. S. e CORNELL, G., **Core Java 2, Volume II – Advanced Features**, Sun Microsystems Press, Prentice Hall, 2000.

ITS TU-Delft, Computer Graphics & CAD/CAM, **http://www.cg.its.tudelft.nl/**, 2000.

KERN, V. M. e BOHN, J. H., STEP database for product data exchange, **I International Congress of Industrial Engineering**, Setembro 4-7, 1995.

KERN, V. M., BOHN, J. H. e BARCIA, R. M., The building of informatioon models in STEP, **II International Congress of Industrial Engineering**, Outubro 7-10, 1996.

KRAUSE, F. –L.; KIMURA, F.; KJELLBERG, T.; LU, S. C. –Y.; ALTING, L.; ELMARAGHY, H. A.; EVERSHEIN, W.; IWATA, K.; SUH, N. P.; TIPNIS, V. A.; WECK, M.; A. C. H.; Product Modeling; **Anais do CIRP**; vol. 42/2/1993; pg. 695-705; 1993.

KRISHNASWAMY, G. M e ELSHENNAWY, A. K., Intelligent Concurrent Engineering Enviroment, **Computers and Industrial Engineering**, vol. 25, n° 1-4, pp. 321-324, 1993.

KUFFNER, T. A. e ULLMAN, D. G., The Information Requests of Mechanical Design Engineers, **Design Studies**, Vol. 12, No. 1, pp. 42-50, 1991.

LEEUWEN, J. P. V. e WAGTER, H., Architectural design by features, **Anais da 7° International Conferce on computer Aided Architectural Design Futures**, pg. 97-115, 1997.

LEIBL, P.; HUNDAL, M. e HOEHNE, G., Cost calculation with a feature based CAD system using modules for calculation, comparison and forecast, **Journal of Engeneering Design**, Vol. 10, no. 1, pg. 93-102, 1999.

LIMA, C. P., **Modelamento baseado em *features* em um conceito de projeto para fabricação e montagem**, Dissertação de Mestrado em Engenharia Mecânica, Depto. de Engenharia Mecânica, Universidade Federal de Santa Catarina, 1994.

LINHARES, J. C., **Modelamento de dados para o desenvolvimento e representação de peças: estudo de casos**, Dissertação de Mestrado em Engenharia Mecânica, Depto. Engenharia Mecânica, Universidade Federal de Santa Catarina, 2000.

LUBY, S. C.; DIXON, J. R. E SIMMONS, M. K.; Creating and Using A Features Data Base; **Computers in Mechanical Engineering**; pg. 25-33; November; 1986.

MARTINO, T.; FALCIDIENO, B. e HABINGER, S., Design and engineering process integration through a multiple view intermediate modeller in a distributed object-oriented system environment. **Computer-Aided Design**, Vol. 30, No. 6, pg. 437-452, 1998.

MAZIERO, N. L, **Um sistema computacional inteligente de suporte ao projeto, manufatura e montagem de peças baseado em features: uma abordagem com sistemas especialistas**, Tese de Doutorado em Engenharia Mecânica, Depto. Engenharia Mecânica, Universidade Federal de Santa Catarina, 1998.

McMAHON, C. e BROWNE, J, **CAD/CAM Principles, Practice and Manufacturing Manangement**; 2° Edição; Addison Wesley Longman Inc., 1998.

MOTOVALLI, S., CHERAGHI, S. H. e SHAMSAASEF, R., Feature-Based Modeling; an onject oriented approach, **Computers Industrial Engineering**, Vol.33, n. 1-2, pg. 349-352, 1997.

NETO, A. F., FURLAN, J. D. e HIGA, W., **Engenharia da Informação – Metodologia, Técnicas e Ferramentas**, 2° Edição, Editora McGraw – Hill, São Paulo –SP, 1988.

MicroStation Modeler, User Guide, Version 5.5, Bentley Systems, 1996.

OGLIARI, A., **Sistematização da concepção de produtos auxiliada por computador com aplicações no domínio de componentes de plástico injetados**, Tese de doutorado em Engenharia Mecânica, Depto. Engenharia Mecânica, Universidade Federal de Santa Catarina, 1999.

OZAWA, M., CUTKOSKY, M. R. e HOWLEY, B. J., Model sharing among agents in a concurrent product development team, **IFIP Working Group 5.2, Terceiro Workshop KIC-3**, 1998.

OSHUGA, S., Toward intelligent CAD systems, **Computer-Aided Design**, Vol. 21, n.5, pg. 315-337, 1989.

PAHL, G. e BEITZ, W, **Engineering Design – A Systematic Approach**, 2° Edição; Springer, London, 1995.

PROVENZA, F., **Projetista de Máquinas**, Pro-tec, São Paulo - SP, 1985.

ROSA, E.; SILVA, J. C.; SILVA, C. A.; RAMINELLI, L. F.; MOREIRA, N. P.; FARIA, P. O.; POSTAL, R.; VIEIRA, R. S.; **New Modeller – Um ambiente computacional de suporte**

à engenharia simultânea; GRANTE; GRUCON; Departmento de Engenharia Mecânica; Universidade Federal de Santa Catarina, Florianópolis (SC), Brasil, 1992.

ROY, U. e PANAYIL, D., Development of a Feature-Based Rapid Design Environment, **Computer Applied in Engineering Education**, Vol. 5, pg 41-60, 1997.

SAHAI, R. S., Inside Project Bank DGN, programing Project Bank, **MSM - MicroStation Manager**, Vol. 10, n°12, 2000.

SAHAI, R. S., **Teach Yourself MicroStation/J**, Alpha Press, 1° Edição, Sterling, 1999.

SALOMONS, O. W., **Computer support in the design of mechanical products. Constraint specification and satisfactionin machanical feature-based design and manufacturing**, Tese de Doutorado, Universidade de Twente, 1995.

SALOMONS, O. W., VAN HOUTEN, F. J. A. M., KALS, H. J. J., Review of research in feature-based design, **Journal of Manufacturing Systems**, Vol. 12, no. 2, pg. 113-132, 1993.

SCHÜTZER, K., GLOCKNER, C., BATOCCHIO, A., Implementação e teste de um ambiente integrado de projeto baseado em "manufacturing features", **Anais do XV Congresso Brasileiro de Engenharia Mecânica – COBEM**, Novembro 22-26, 1999.

SHAH, J. J., Assessment of Features Technology; **Computer-Aided Design**; vol. 23, n° 5, pg. 331-343, 1991.

SHAH, J. J. e ROGERS, M., Expert Form Feature Modeling Shell, **Computer-Aided Design**, vol. 20, n° 9, pg. 515-524, 1988.

SHAH, J. J. e MÄNTYLÄ, M., **Parametric and Feature-Based CAD/CAM - Concepts, Techniques, Applications**, Wiley Intercience, New York, 1995.

SHAH, J. J. e MATHEW, A., An experimental investigation of the STEP form feature information model, **Computer-Aided Design**, vol. 23, n° 4, pg. 282-296, 1991.

SILVA Jr., A. C. e DIAS, A., Aspects of Concurrent Engeneering Implementation in the CAD/CAM Systems Environment, **Anais do Congresso Flexible Automation and Inteligent Manugfacturing – FAIM**, Vol. 2, pg. 1127-1135, 2000.

SILVA, M. M. e ALLIPRANDINI, D. H.; Análise do processo de desenvolvimento do produto: Estudo de caso em empresas manufatureiras baseado em um modelo referencial para caracterização e diagnóstico; **Anais do IICBGDP**, 2000.

SSEMAKULA, M. E. e SATSANGI, A., Aplication of PDES to CAD/CAPP integration, **Computers Industrial Engineering**, Vol. 18, n. 4, pg. 435-444, 1990.

STALEY, S. M e ANDERSON, D. C., Functional specification for CAD database, **Computer-Aided Design**, Vol. 18, n. 3, 1986.

VALERI, S. G., ROZENFELD, H. e ALLIPRANDINI, D. H.; Análise do processo de desenvolvimento de produtos de uma indústria do setor automobilístico; **Anais do IICBGDP**, 2000.

VIESCAS, J. L., **Microsoft Access 2 for Windows, Guia autorizado Microsoft**, Ed. Makron Books, Rio de Janeiro, 1995.

WANG, M. e WALKER, H., Creation of an intelligent process planning system within the relational DBMS software environment, **Computers in Industry**, vol. 13, pg. 215-228, 1989.

ZAIANE, O. R. e KOPERSKI, K., CMPT 354 Database Systems and Structures, **http://www.cs.sfu.ca/CC/354/zaiane/**, 1998.

ZEID, I.; **CAD/CAM Theory and Practice**; McGraw-Hill, Inc.; New York; 1991.

ANEXO 1

BIBLIOTECA DE FEATURES

Tabela A1.1: *Features* Básicas Elementares

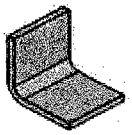
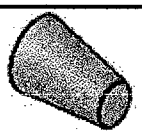
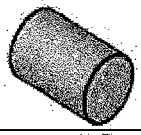
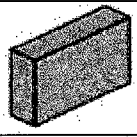
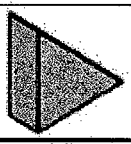
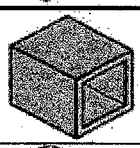
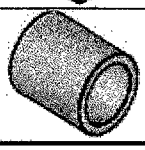
CLASSIFICAÇÃO		FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
				Normalizados	Indep.	Dependentes	
PROJEÇÃO	PERFIS		Perfil L	Lado 1 Lado 2 Ângulo Raio de Arredon. Espessura	Comprimento	-	Peso (Kg/m) Material
	BARRAS		Tronco de cone	Raio maior Ângulo	Comprimento	-	Peso (Kg/m) Material
			Barra cilíndrica	Diâmetro	Comprimento	-	Peso (Kg/m) Material
			Barra retangular	Largura Espessura	Comprimento	-	Peso (Kg/m) Material
			Barra triangular	Espessura	Lado 1 Lado 2 Ângulo	-	Peso (Kg/m) Material
	TUBOS		Tubo retangular	Largura Altura Espessura	Comprimento	-	Peso (Kg/m) Material
			Tubo cilíndrico	Diâmetro Espessura	Comprimento	-	Peso (Kg/m) Material

Tabela A1.2: *Features* Adicionais Elementares

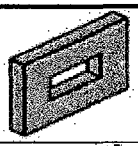
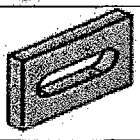
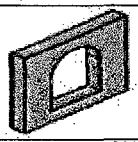
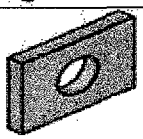
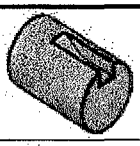
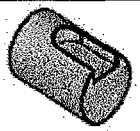
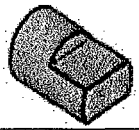
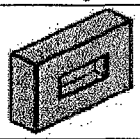
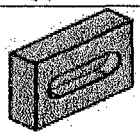
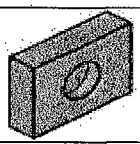
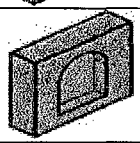
CLASSIFICAÇÃO			FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
					Normalizados	Indep.	Dependente	
CORTE	RETO	FUROS		Furo retangular	-	Largura Altura	Profundidade	Processo Ferramenta
				Furo Oblongo	Diâmetro	Extensão (Distância entre centros)	Profundidade	Processo Ferramenta
				Furo U	Raio	Comprimento	Profundidade	Processo Ferramenta
				Furo circular	Diâmetro	-	Profundidade	Processo Ferramenta
		CANAIS RETOS		Rasgo para chaveta deslizante	Raio da fresa Profundidade Comprimento	-	-	Processo Ferramenta
				Rasgo para chaveta reta	Raio da fresa Profundidade Comprimento	-	-	Processo Ferramenta
		REBAIXOS E ENTELHES		Espiga	-	Espessura Comprimento	Diâmetro	Processo Ferramenta
				Rebaixo retangular	-	Largura Comprimento Profundidade	-	Processo Ferramenta
				Rebaixo O	Raio	Distância centros Profundidade	-	Processo Ferramenta
				Furo cego	Diâmetro	Profundidade	-	Processo Ferramenta
				Rebaixo U	Raio	Comprimento Profundidade	-	Processo Ferramenta

Tabela A1.3: *Features* Adicionais Elementares (Continuação 1)

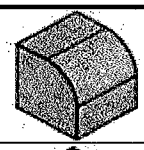
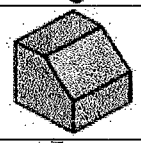
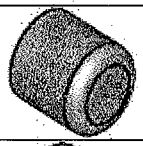
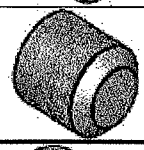
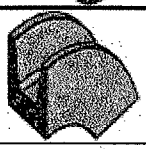
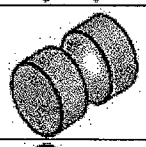
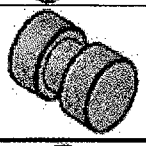
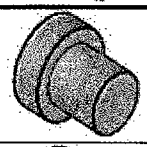
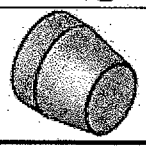
CLASSIFICAÇÃO			FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
					Normalizados	Indep.	Dependente	
CORTE	CAMINHO LIVRE	CHANFROS E ARREDONDAMENTOS EM QUINAS		Arredonda - mento em quina reta	Raio	-	Comprimento	Processo Ferramenta
				Chanfro em quina reta	Lado 1 Lado 2	-	Comprimento Ângulo	Processo Ferramenta
				Arredonda - mento em quina circular	Raio	-	Comprimento	Processo Ferramenta
				Chanfro em quina circular	Lado 1 Lado 2	-	Comprimento Ângulo	Processo Ferramenta
		CANAIS E RASGOS CIRCULARES		Canal para correia	Ângulo Altura Base Raio de arr.	-	-	Processo Ferramenta
				Canal circular meia lua	Diâmetro	-	-	Processo Ferramenta
				Canal retangular	Profundidade Largura	-	-	Processo Ferramenta
		ESCALONAMENTOS E ALARGAMENTOS		Escalona - mento Cilíndrico	Raio Altura Largura	-	-	Processo Ferramenta
				Escalona - mento Cônico	Raio maior Ângulo Comprimento	-	-	Processo Ferramenta

Tabela A1.4: *Features* Adicionais Elementares (Continuação 2)

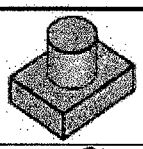
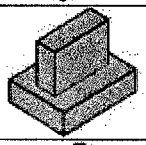
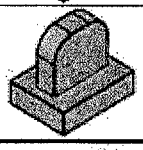
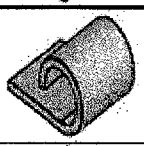
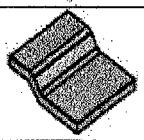
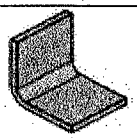
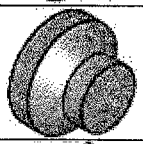
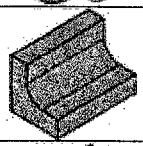
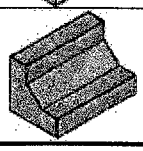
CLASSIFICAÇÃO			FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
					Normalizados	Indep.	Dependente	
PROTUSÃO	RETA	PROTUSÕES E SALIÊNCIAS RETAS		Protusão circular	-	Diâmetro Altura	-	Processo Ferramenta
				Protusão Retangular	-	Largura Altura Comprimento	-	Processo Ferramenta
				Orelha U	-	Comprimento Raio Largura Espessura	-	Processo Ferramenta
	CAMINHO LIVRE	DOBRAS		Dobra circular	-	Folga Raio Altura	Espessura	Processo Ferramenta
				Dobra off-set	Raio Arredonda-mento	Aba Superior Ângulo Altura	Espessura	Processo Ferramenta
				Dobra L	Raio Arredonda-mento	Lado 1 Lado 2 Ângulo	Espessura	Processo Ferramenta
		SALIÊNCIAS CIRCULARES		Saliência circular meia lua	-	Raio	-	Processo Ferramenta
		CHANFROS E ARREDONDAMENTOS EM CANTOS		Arredonda - mento em canto circular	Raio	-	Comprimento	Processo Ferramenta
				Chanfro em canto circular	Lado 1 Lado 2	-	Comprimento Ângulo	Processo Ferramenta
				Arredonda - mento em canto reto	Raio	-	Comprimento	Processo Ferramenta
				Chanfro em canto reto	Lado 1 Lado 2	-	Comprimento Ângulo	Processo Ferramenta

Tabela A1.5: *Features* Adicionais Parciais

CLASSIFICAÇÃO			FORMA GEOMÉTRICA	NOME DA *FEATURE*	PARÂMETROS DIMENSIONAIS			ATRIBUTOS TECNOLÓGICOS
					Normalizados	Indep.	Dependente	
PROTUSÃO	CAMINHO LIVRE	PIONTA DE EIXOS		Ponta do eixo central da Betoneira 120L	Cone 1 Cone 3 Ângulo Raio Maior	Cone 2 Desnível	-	Processo Ferramenta
	RETA	RAIOS DE POLIAS		Raio de polia padrão	-	Raio arr. Superior Raio arr. Inferior Raio roda Largura	Raio cubo	Processo Ferramenta
CORTE	RETO	FUROS ESPECIAIS		Furo chavetado	Abertura Altura Diâmetro	-	Profundidade	Processo Ferramenta

ANEXO 2

PROTOCOLO DE COMUNICAÇÃO DA INTERFACE CAD-RDBMS

Tabela A2.1: Protocolo de comunicação da Interface CAD-RDBMS

PARÂMETROS	INFORMAÇÕES ASSOCIADA	VALORES PADRÕES
FEATURE	O nome da feature	PROJECTION - Projeção CUT - Corte reto PROTRUSION - Protusão
CELL	O nome do perfil 2D parametrizado	-
THICK	Espessura da parede gerada.	-
ANGLE	Ângulo de varredura	-
DIRECTION	Direção da operação construtiva de varredura.	0 - Para frente da face de referência 1 - Para trás da face de referência 2 - Em ambas as direções
FW_T	Tipo de operação da varredura para frente da face de referência	0 - Cego 2 - Passante 3 - Até a próxima face
RV_T	Tipo de operação de varredura para trás da face de referência	0 - Cego 2 - Passante 3 - Até a próxima face
FW_D	Distância de varredura para frente da face.	-
RV_D	Distância de varredura para trás da face	-
NP	Número de parâmetros existentes no perfil parametrizado. O valor desta parâmetro deve ser declarado antes de qualquer PARAM.	-
PARAM	Valor numérico de um parâmetro da célula. Se o perfil tiver mais de um (01) parâmetro, deve-se repetir o rótulo PARAM. A ordem dos PARAM da esquerda para a direita deve ser a mesma ordem dos parâmetros no perfil 2D, no sentido anti-horário, a partir do ponto de âncora do perfil.	-

Obs. 1: A não ser com relação aos parâmetros "NP" e "PARAM", a ordem de colocação dos mesmos não interfere no resultado.

Obs. 2: Quando algum parâmetro não se aplicar ou não for necessário para construção da forma geométrica da *feature*, ele não precisa ser declarado no protocolo

Exemplos de aplicação:

- Protocolo para construção de uma "Barra Retangular" através do uso da Feature Genérica: Projeção e de um perfil 2D parametrizado de um retângulo, denominado: 02. A barra tem 130 mm de comprimento, 4.8 mm de espessura e 28.0 mm de largura. A Figura A2.1 ilustra o perfil 2D usado, a seta indica a ordem de inserção dos "PARAM" declarados: *FEATURE=PROJECTION CELL=02 FW_D=130.0 DIRECTION=0 NP=2 PARAM=4.8 PARAM=28.0*

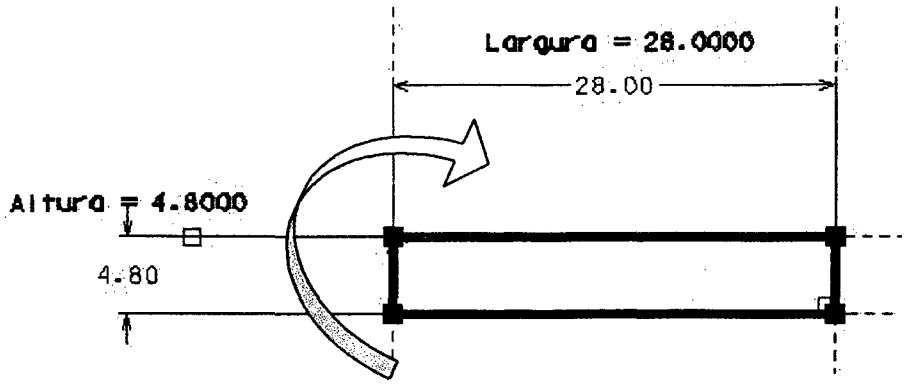


Figura A2.1: Perfil retangular

- Protocolo para construção de um "Furo O", através do uso da Feature Genérica: Corte reto e de um perfil 2D parametrizado de um retângulo com as pontas arredondadas, denominado: 05. O furo tem 4.8 mm de profundidade, 9.0 mm de diâmetro nas pontas arredondadas e 10.0 mm de distancia entre os centros das pontas A Figura A2.2 ilustra o perfil 2D usado, a seta indica a ordem de inserção dos "PARAM" declarados: *FEATURE=CUT CELL=05 DIRECTION=1 FW_T=0 FW_D=4.8 NP=2 PARAM=9.0 PARAM=10.0*

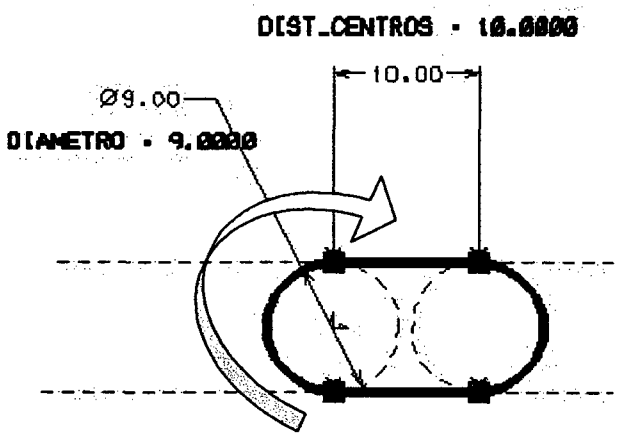


Figura A2.2: Perfil O

ANEXO 3

CÓDIGO FONTE DO MÓDULO 1 DA INTERFACE CAD-RDBMS, ESCRITO EM MDL

THROWSFEATURESMDL.MC

```
/*----------
 #
 # Arquivos "include"
 #
 ----------*/
#include   <string.h>
#include   <mscons.h>
#include   <assert.h>
#include   <mdl.h>
#include   <stdlib.h>
#include   <mselems.h>
#include   <tcb.h>
#include   <mssolid.h>

/*----------
 #
 # Arquivos de definição de funções
 #
 ----------*/
#include   <mscnv.fdf>
#include   <msdialog.fdf>
#include   <mselemen.fdf>
#include   <mselmdsc.fdf>
#include   <msfeatur.fdf>
#include   <mslocate.fdf>
#include   <msmodapi.fdf>
#include   <msmodel.fdf>
#include   <msmodlib.fdf>
#include   <msrmatrx.fdf>
#include   <msrsrc.fdf>
#include   <msstate.fdf>
#include   <mssolid.fdf>

/*----------
 #
 # Variávei globais
 #
 ----------*/
/* — Referentes à célula parametrizada — */
char cellName[3];          /*O nome da célula parametrizada deve ter no máximo 2 caracteres*/
double parametros[10];     /*A célula parametrizada deve ter no máximo 10 parâmetros*/
MSElementDescr *cellEDP;

/* — Referentes às features — */
char feature[15];
double thickness=NULL, draftAngle=NULL, ddcellParam[10], fwDistance=NULL, rvDistance=NULL;
double rightDistance=NULL, leftDistance=NULL, startRadius=NULL, endRadius=NULL;
int direction=NULL, fwThrough=NULL, rvThrough=NULL, numParams;
MSElementDescr *addFeatureEDP=NULL;

/* — Referente à métodos construtivos — */
Dpoint3d points[10];
int contPoints=0;

/* — Referentes à árvore de features — */
MSElementDescr *evaluatedTreeEDP=NULL, *treeEDP=NULL;


/*----------
 #
 # Método de finalização
 #
 ----------*/
Private void finish(void)
  {
  if ( cellEDP )
```

```
    mdlElmdscr_freeAll (&cellEDP);

  mdlState_startDefaultCommand ();
  }
/*-------------------------------
 #
 # Captura a matriz de transformação da célula
 #
 -------------------------------*/
Private void getCell_transform(Dpoint3d *pt, int view, Transform *tMatrix)
  {
  RotMatrix rMatrix, angle_rMatrix;

  /* -- matriz de rotação da vista [RotV] -- */
  if ( mdlCell_isPointCell (&cellEDP->el) )
    {
    mdlRMatrix_getIdentity (&rMatrix);
    }
  else
    {
    mdlRMatrix_fromView (&rMatrix, view, TRUE);
    mdlRMatrix_invert (&rMatrix, &rMatrix);
    }
  /* -- matriz de angulo ativo de rotação [RotAA] -- */
  mdlRMatrix_rotate (&angle_rMatrix, NULL,
                fc_zero, fc_zero, (tcb->actangle)*fc_piover180);

  /* -- matris produto [RotV][RotAA] -- */
  mdlRMatrix_multiply (&rMatrix, &rMatrix, &angle_rMatrix);

  /* -- conversão da matriz produto para Transformada -- */
  mdlTMatrix_getIdentity (tMatrix);
  mdlTMatrix_fromRMatrix (tMatrix, &rMatrix);

  /* -- pós-multiplica [Scale] => [RotV][RotAA][Scale] -- */
  mdlTMatrix_scale (tMatrix, tMatrix,
                tcb->xactscle, tcb->yactscle, tcb->zactscle);
  }
/*-------------------------------

 # Atualização dinâmica da matriz de transformação

 -------------------------------*/
Private void dynamicCell(Dpoint3d *pt, int view)
  {
  Transform tMatrix;

  getCell_transform (pt, view, &tMatrix);

  /* -- pré-multiplica [Trans] => [Trans][RotV][RotAA][Scale] -- */
  mdlTMatrix_setTranslation (&tMatrix, pt);

  mdlElement_transform (dgnBuf, dgnBuf, &tMatrix);
  }
/*-------------------------------
 #
 # Inicia a construção das features adicionais com a
 # captura da árvore de features da feature basica
 #
 -------------------------------*/
Private void startAddFeature(void)
  {
  ULong  filePosition;
  int status;

  filePosition = mdlElement_getFilePos(FILEPOS_CURRENT, MASTERFILE);
  mdlElmdscr_read (&addFeatureEDP, filePosition, MASTERFILE, FALSE, NULL);
```

```
  /* — Captura da árvore de features do elemento selecionado — */
  if (SUCCESS == (status = mdlFeature_treeFromElementAndFilePos(&treeEDP, NULL, addFeatureEDP,
                    MASTERFILE, filePosition)))
    {
    /* — Coloca o descritor no buffer dinâmico do microstation —*/
    mdlDynamic_setElmDescr (cellEDP);
    /* — Especifica a função de atualização dinâmica —*/
    mdlState_dynamicUpdate (dynamicCell, FALSE);
    }
  }
/*-----------------------
 #
 # Zera todas as variáveis globais
 #
 -----------------------*/
Private void nullVars()
  {
  int cont;

  for(cont=0; cont<3; cont++)
    cellName[cont]=NULL;

  for(cont=0; cont<10; cont++)
    parametros[cont]=NULL;

  for(cont=0; cont<15; cont++)
    feature[cont]=NULL;

  contPoints=0;
  rightDistance=NULL;
  leftDistance=NULL;
  startRadius=NULL;
  endRadius=NULL;

  cellEDP=NULL;
  thickness=NULL;
  draftAngle=NULL;
  fwDistance=NULL;
  rvDistance=NULL;
  direction=NULL;
  fwThrough=NULL;
  rvThrough=NULL;
  numParams;
  addFeatureEDP=NULL;
  evaluatedTreeEDP=NULL;
  treeEDP=NULL;
  }
 /*-------------------------------------
 #
 # Atribui os valores das variáveis globais determinados
 # na String entrada através do keyin
 #
 -------------------------------------*/
Private void setVars(char paramString[])
  {
  char *p;
  int contDDCellParam=0;

  p = strtok(paramString, "=");
  do
    {
    if(strcmp(p, "FEATURE") == 0)
      {
      p = strtok('\0', " ");
      strcpy(feature, p);
      }
```

```
    else if(strcmp(p, "CELL") == 0)
      {
      p = strtok('\0', " ");
      strcpy(cellName, p);
      }
    else if(strcmp(p,"THICK")==0)
      {
      p = strtok('\0', " ");
      thickness = atof(p);
      }
    else if(strcmp(p, "FW_D")==0)
      {
      p = strtok('\0', " ");
      fwDistance = atof(p);
      }
    else if(strcmp(p, "RV_D")==0)
      {
      p = strtok('\0', " ");
      rvDistance = atof(p);
      }
    else if(strcmp(p, "ANGLE")==0)
      {
      p = strtok('\0', " ");
      draftAngle = atof(p);
      }
    else if(strcmp(p,"DIRECTION")==0)
      {
      p = strtok('\0', " ");
      direction = atoi(p);
      }
    else if(strcmp(p, "FW_T")==0)
      {
      p = strtok('\0', " ");
      fwThrough = atoi(p);
      }
    else if(strcmp(p, "RV_T")==0)
      {
      p = strtok('\0', " ");
      rvThrough = atoi(p);
      }
    else if(strcmp(p, "PARAM")==0)
      {
      p = strtok('\0', " ");
      parametros[contDDCellParam] = atof(p);
      ++contDDCellParam;
      }
    else if(strcmp(p, "NP")==0)
      {
      p = strtok('\0', " ");
      numParams = atoi(p);
      }
    p = strtok('\0', "=");
    } while(p);
  }
/*-------------------------------
 #
 # Construção da feature Parametric Projection
 #
 -------------------------------*/
Private void makeProjection(Dpoint3d *pntP, int view)
  {
  MSElementDescr *nodeProjectionEDP=NULL;
  RotMatrix rotMatrix;
  Transform transMatrix;
  int status;

  /* --- Captura da matriz de rotação a partir da orientação da célula --- */
```

```
  getCell_transform(pntP, view, &transMatrix);
  mdlRMatrix_fromTMatrix(&rotMatrix, &transMatrix);

  /* — Conversão do Data Point pata Mater Units */
  mdlCnv_UORToMaster (&pntP->x, pntP->x, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->y, pntP->y, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->z, pntP->z, MASTERFILE);

  mdlFeature_createProjectionNode(&treeEDP, &nodeProjectionEDP, cellEDP, cellName, thickness, fwDistance,
draftAngle,
                        pntP, &rotMatrix, MASTERFILE);

  /* — Tratamento das dimensões da célula — */
  mdlFeature_addCellDefLinkage(&treeEDP, &nodeProjectionEDP, cellName, numParams, parametros, TRUE);

  /* — Avaliação da árvore de features treeEdP — */
  if ((status = mdlFeature_evaluateTree (NULL, &evaluatedTreeEDP, treeEDP, MASTERFILE, -1,
    -1, FALSE, mdlFeature_checkNodeErrorHandler, treeEDP)) == SUCCESS)
    {
    /* — Adição da árvore de features para o arquivo de projeto — */
    /*edP é um dercritor de elemento que contém a árvore de features já avaliada e sua geometria*/
    mdlElmdscr_add (evaluatedTreeEDP);
    /* — Exibição da árvore de features — */
    mdlElmdscr_display (evaluatedTreeEDP, MASTERFILE, NORMALDRAW);
    /* — Liberação de toda memória alocada pelo descritor de elemento — */
    mdlElmdscr_freeAll (&evaluatedTreeEDP);
    }
  else
    printf ("Error %d\n", status);
  if (treeEDP)
    mdlElmdscr_freeAll (&treeEDP);

  finish();
  }
/*————————————————
 #
 # Construção da feature Protrusion
 #
 ————————————————*/
Private void protrusionAccept(Dpoint3d *pntP, int view)
  {
  MSElementDescr *nodeProtrusionEDP=NULL;
  RotMatrix rotMatrix;
  Transform transMatrix;
  int status;

  /* — Coloca o descritor no buffer dinâmico do microstation —*/
  mdlDynamic_setElmDescr (cellEDP);

  /* — Especifica a função de atualização dinâmica —*/
  mdlState_dynamicUpdate (dynamicCell, FALSE);

  /* — Captura da matriz de rotação a partir da orientação da célula — */
  getCell_transform(pntP, view, &transMatrix);
  mdlRMatrix_fromTMatrix(&rotMatrix, &transMatrix);

  /* — Conversão do Data Point pata Mater Units */
  mdlCnv_UORToMaster (&pntP->x, pntP->x, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->y, pntP->y, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->z, pntP->z, MASTERFILE);

  /* — Adiciona cut node to the feature tree — */
  mdlFeature_addProtrusionNode(&treeEDP, &nodeProtrusionEDP, cellEDP, cellName, thickness, fwDistance,
rvDistance,
              draftAngle, direction, fwThrough, rvThrough, pntP, &rotMatrix, MASTERFILE);

  /* — Tratamento das dimensões da célula — */
```

```
    mdlFeature_addCellDefLinkage(&treeEDP, &nodeProtrusionEDP, cellName, numParams, parametros,
TRUE);

  /* — Chama "evaluate tree" — */
  if (SUCCESS == (status = mdlFeature_evaluateTree(NULL, &evaluatedTreeEDP, treeEDP, MASTERFILE, -1,
                    -1, FALSE, mdlFeature_checkNodeErrorHandler, treeEDP)))
    {
    mdlModeler_incrementalBodyDisplay (addFeatureEDP, evaluatedTreeEDP,
                        mdlElement_getFilePos (FILEPOS_CURRENT, NULL), HILITE);
    mdlModeler_rewriteExt (&evaluatedTreeEDP, 0L, MASTERFILE, TRUE, TRUE, TRUE);

    mdlElmdscr_freeAll(&addFeatureEDP);
    }
  if(treeEDP)
    mdlElmdscr_freeAll(&treeEDP);
  if(evaluatedTreeEDP)
    mdlElmdscr_freeAll(&evaluatedTreeEDP);

  finish();
  }
/*-----------------
 #
 # Construção da feature Cut
 #
 -----------------*/
Private void cutAccept(Dpoint3d *pntP, int view)
  {
  MSElementDescr *nodeCutEDP=NULL;
  RotMatrix rotMatrix;
  Transform transMatrix;
  int status;

  /* — Captura da matriz de rotação a partir da orientação da célula — */
  getCell_transform(pntP, view, &transMatrix);
  mdlRMatrix_fromTMatrix(&rotMatrix, &transMatrix);

  /* — Conversão do Data Point pata Mater Units */
  mdlCnv_UORToMaster (&pntP->x, pntP->x, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->y, pntP->y, MASTERFILE);
  mdlCnv_UORToMaster (&pntP->z, pntP->z, MASTERFILE);

  /* — Adiciona cut node to the feature tree — */
  mdlFeature_addCutNode(&treeEDP, &nodeCutEDP, cellEDP, cellName, thickness, fwDistance, rvDistance,
              TRUE, direction, fwThrough, rvThrough, draftAngle, pntP, &rotMatrix, MASTERFILE);

  /* — Tratamento das dimensões da célula — */
    mdlFeature_addCellDefLinkage(&treeEDP, &nodeCutEDP, cellName, numParams, parametros, TRUE);

  /* — Chama "evaluate tree" — */
  if (SUCCESS == (status = mdlFeature_evaluateTree(NULL, &evaluatedTreeEDP, treeEDP, MASTERFILE, -1,
                    -1, FALSE, mdlFeature_checkNodeErrorHandler, treeEDP)))
    {
    mdlModeler_incrementalBodyDisplay (addFeatureEDP, evaluatedTreeEDP,
                        mdlElement_getFilePos (FILEPOS_CURRENT, NULL), HILITE);
    mdlModeler_rewriteExt (&evaluatedTreeEDP, 0L, MASTERFILE, TRUE, TRUE, TRUE);

    mdlElmdscr_freeAll(&addFeatureEDP);
    }
  if(treeEDP)
    mdlElmdscr_freeAll(&treeEDP);
  if(evaluatedTreeEDP)
    mdlElmdscr_freeAll(&evaluatedTreeEDP);

  finish();
  }
/*-------------------------------
 #
```

```
# Função que passa os parâmetros iniciais para a
# função de construção de uma feature primitiva
#
-------------------------------------*/
cmdName void makeFeature(char p[])
  {
  cellEDP = NULL;

  nullVars();

  setVars(p);

  /* — Captura do elemento descritor da célula — */
  mdlCell_getElmDscr(&cellEDP, NULL, NULL, NULL, NULL, NULL, NULL, 0, 1, cellName);

  /* — Especifica a função para o Data Point —*/
  if(strcmp(feature, "PROJECTION")==0)
    {
    mdlState_startPrimitive (makeProjection, finish, NULL, NULL);

    /* — Coloca o descritor no buffer dinâmico do microstation —*/
    mdlDynamic_setElmDescr (cellEDP);

    /* — Especifica a função de atualização dinâmica —*/
    mdlState_dynamicUpdate (dynamicCell, FALSE);
    }
    /* — Especifica a função para o Data Point —*/
  else if(strcmp(feature, "PROTRUSION")==0)
    mdlState_startModifyCommand (finish, protrusionAccept, NULL, startAddFeature, NULL, NULL, NULL,
FALSE, 0);
  else if(strcmp(feature, "CUT")==0)
    mdlState_startModifyCommand (finish, cutAccept, NULL, startAddFeature, NULL, NULL, NULL, FALSE, 0);
  }
```

THROWSFEATURESMDL.MKE ("MAKE FILE")

```
appName = ThrowsFeaturesMDL

%include mdl.mki

#----------------------------------------------------------
# Define macros específicas para este exemplo
#----------------------------------------------------------
baseDir = ./

#privateInc e altIncs são variáveis de mdl.mki
#eles são usados juntos com o flag -i<dir> para especificar uma lista de diretórios
#procurados para "include"
privateInc = $(MSMODELER)mdl/examples/modapi/include \

altIncs    + -i$(MSMODELER)mdl/include \
             -i$(MSMODELER)mdl/examples/modapi/include \

#----------------------------------------------------------
#      Object Files
#----------------------------------------------------------
testObjs = $(o)$(appName).mo \
           $(MSMODELER)mdl/library/msmodapi.ml \
           $(MSMODELER)mdl/library/modellib.dlo \
           $(MSMODELER)mdl/library/modeldlm.dlo

testRscs = $(o)$(appName).mp

#----------------------------------------------------------
# Compile the MDL source object file using mcomp
#----------------------------------------------------------
```

```
$(o)$(appName).mo       : $(baseDir)$(appName).mc

#-------------------------------------------------------
# Link MDL program file from object (.mo) using rlink
#-------------------------------------------------------
$(o)$(appName).mp       : $(testObjs)
        $(msg)
        > $(o)temp.cmd
        -a$@
        -s7000
        $(linkOpts)
        $(testObjs)
        <
        $(MlinkCmd) @$(o)temp.cmd
        ~time

#-------------------------------------------------------
# Create app file (.ma) from program file and it's resources using rlib
#-------------------------------------------------------
$(baseDir)$(appName).ma         : $(testRscs)
        $(msg)
        > $(o)temp.cmd
        -o$@
        $(testRscs)
        <
        $(RLibCmd) @$(o)temp.cmd
        ~time
```