UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Contribuições para o Desenvolvimento de um Modelador Baseado em *Form Features* com Interface STEP - ISO 10303

ANTONIO ÁLVARO DE ASSIS MOURA

ORIENTADOR: PROF. DR.-ING. KLAUS SCHÜTZER

SANTA BÁRBARA D'OESTE

2003

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

Contribuições para o Desenvolvimento de um Modelador Baseado em *Form Features* com Interface STEP - ISO 10303

Antonio Álvaro de Assis Moura

Orientador: Prof. Dr.-Ing. Klaus Schützer

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia de Produção, da Faculdade de Engenharia Arquitetura e Urbanismo, da Universidade Metodista de Piracicaba - UNIMEP

Santa Bárbara d'Oeste

2003

Contribuições para o Desenvolvimento de um Modelador Baseado em *Form Features* com Interface STEP - ISO 10303

Antonio Álvaro de Assis Moura

Dissertação de Mestrado defendida e aprovada, em 09 de maio de 2003, pela Banca Examinadora constituída pelos Professores:

Prof. Dr.-Ing. Klaus Schützer, Presidente
PPGEP - UNIMEP

Prof. Dr. Jonas de Carvalho
EESC - USP

Profa. Dra. Tereza Gonçalves Kirner
UNIMEP

À família Fante.

Sumário

Lista de Figuras

Lista de Siglas

AAM	Application Activity Model
AIM	Application Interpreted Model
AP	Application Protocol
API	Application Programming Interface
ARM	Application Reference Model
CAD	Computer Aided Design
CAPP	Computer Aided Process Planning
CASE	Computer Aided Software Engineering
CAx	Computer Aided x (Utilizado para indicar qualquer sistema auxiliado por computador)
CORBA/IDL	Common Object Request Broker Architecture/Interface Definition Language
DIN	Deutsches Institut für Normung
HTML	Hypertext Markup Language
IGES	Initial Graphics Exchange Specification
ISO	International Organization for Standardization
MFC	Microsoft Foundation Classes
OMG	Object Management Group
OMT	Object Modeling Technique
PDT	Product Data Technology
SCL	Step Class Library
SDAI	Standard Data Access Interface

SET	Standard d’exchange et de Transfer
SQL	Structured Query Language
STEP	Standard for the Exchange of Product Model Data
UG	Unigraphics (CAD Software by Unigraphics Solutions)
UML	Unified Modeling Language

Lista de Institutos e Laboratórios

NIST	National Institute of Standards and Technology
UNIMEP	Universidade Metodista de Piracicaba
VDA FS	Verband der deutschen Automobilindustrie Flächenschnittstelle
PTW	Institut für Produktionsmanagement, Technologie und Werkzeugmaschinen
DiK	Institut für Datenverarbeitung in der Konstruktion
ISO	International Organization for Standartization
SCPM	Laboratório de Sistemas Computacionais para Projeto e Manufatura - Faculdade de Engenharia, Arquitetura e Urbanismo - Universidade Metodista de Piracicaba

Resumo

MOURA, Antonio Álvaro de Assis. *Contribuições para o Desenvolvimento de um Modelador Baseado em Form Features com Interface STEP - ISO 10303.* 2003. 139 p. Dissertação de Mestrado - Faculdade de Engenharia Arquitetura e Urbanismo, Universidade Metodista de Piracicaba, Santa Bárbara d'Oeste.

Esta dissertação aborda a integração entre as fases do ciclo de desenvolvimento do produto, de modo a preservar a qualidade do produto final, eliminar o retrabalho e as interferências de interpretação nas transferências de informações do modelo entre as diversas ferramentas utilizadas.

Inicialmente, são apresentadas as razões que levam à busca desta integração, seus benefícios e os problemas oriundos das perdas de informação no processo de desenvolvimento do produto. São também discutidas as tecnologias disponíveis para esta integração, suas facilidades e os problemas enfrentados neste desenvolvimento.

Como resultado deste projeto de pesquisa serão apresentadas contribuições para um protótipo de um modelador baseado na tecnologia de *form features* que armazena as informações do modelo, dentro dos critérios estabelecidos pelas normas STEP. As informações do modelo registradas nesta forma são passíveis de serem recuperadas nas próximas etapas do ciclo de desenvolvimento do produto.

Todo o desenvolvimento deste protótipo foi executado dentro do paradigma da orientação a objetos. O modelador e o banco de dados, que recebe as informações do modelo, são orientados a objetos. Como ferramenta para criação do software foram utilizadas as padronizações do UML, de modo a se obter uma documentação ampla e detalhada.

Palavras-chave: STEP; *Form Features*; Orientação a Objetos, Projeto Baseado em *features*, Integração CAD/CAM/CAPP.

Abstract

MOURA, Antonio Álvaro de Assis. *Contribuições para o Desenvolvimento de um Modelador Baseado em Form Features com Interface STEP - ISO 10303.* 2003. 139 p. Dissertação de Mestrado - Faculdade de Engenharia Arquitetura e Urbanismo, Universidade Metodista de Piracicaba, Santa Bárbara d'Oeste.

This dissertation is about the integration among all phases of product development cycle, objecting to preserve the final product quality; eliminate the re-work and the interpretation interferences in the transfer of the model information among several tools used.

Initially, are showing the reasons why this integration is searched, these benefits and the problems of loss information in the product development process. Also, are discussing the technologies used in these integrations, these facilities and problems.

As result of this research project is show advances in a modeling prototype based on *features* technology, which save all model information, regarding the STEP criteria. The information model save in this form can be recover in the next phases of product development cycle.

The development of this prototype is done by the oriented-object paradigm, as the database, which will receive the model information. The tool used to create the software prototype follows the UML standardization, by what is possible to do a large and detail documentation.

Keywords: STEP; Form Features; Object-Orientation, Design by features, CAD/CAM/CAPP Integration.

1 Introdução

O processo de criação inicia-se com um modelo mental, passa pela elaboração de um modelo e depois para a execução concreta, física.

Tome-se como exemplo, a criação de uma música. O artista, com capacidade de criação musical, inicia o processo de criação musical com um modelo mental, para então executar a música em um instrumento, ouvi-la, burilá-la até atingir uma maturação satisfatória. Após isto, ou mesmo durante o processo de criação, o artista vai armazenando as informações da música, ou anotando em uma partitura, ou gravando o som do instrumento.

Com a documentação da música elaborada de modo adequado, é possível reproduzi-la exatamente da forma como foi criada por qualquer músico em qualquer lugar, caracterizando assim o objetivo pretendido de perpetuar a criação.

Neste processo, podem ser destacadas algumas etapas principais: a utilização de um modelo para a criação, o registro das informações do modelo e a recuperação das informações para a reprodução.

No caso da música, o modelo mental é um campo de estudo bastante complexo e incipiente, o que não ocorre nos processos de engenharia, nos quais os modelos utilizados, sejam eles concretos ou abstratos, digitais ou analógicos são objetos de profundos estudos.

Já na parte de documentação e recuperação da informação, a música está em estágio bastante avançado. A utilização de partituras com padronização das notas e tempos data de cinco séculos. Uma partitura de Bach, elaborada em 1700, pode hoje ser reproduzida sem maiores dificuldades. Neste caso, tem-se, de modo bastante claro, o registro e a recuperação da informação do criador durante um longo período de tempo.

Da mesma forma que em um processo de criação artística, em um projeto de um produto de engenharia tem-se a utilização de um modelo e o registro de suas informações para a execução do produto real.

Em um projeto de engenharia, o objetivo final é, em última instância, a execução concreta, física, do produto. Durante a fase de projeto, são levados em conta inúmeros fatores como ergonomia, custo, segurança, manutenção, tempo de execução, etc. Terminada esta fase, ou seja, tendo sido concluído o projeto, se faz necessária a execução concreta do produto, sem a qual não se justifica o projeto.

A elaboração do projeto tem tido desenvolvimento constante; várias técnicas foram desenvolvidas para aprimorar o projeto e assegurar a execução do produto, sendo que a utilização de modelos sempre foi a base deste desenvolvimento.

O projeto de um produto de engenharia está tão intimamente ligado à criação de um modelo que, às vezes, os termos modelagem e projeto se confundem. De fato, o projeto de um produto pode ser resumido à criação de um modelo abstrato que possa ser devidamente documentado e que sirva para a execução concreta do produto.

O desenvolvimento das técnicas de modelagem, tanto abstratas como concretas, digitais ou analógicas, tem servido também ao desenvolvimento do projeto de engenharia. Os incrementos tecnológicos matemáticos também são aplicados nos modelos matemáticos utilizados no projeto.

Mais recentemente, a criação de modelos digitais em sistemas computacionais ampliou de tal maneira a complexidade dos modelos que alterou profundamente o modo de se pensar o projeto de engenharia. Tanto que a elaboração de um modelo é feita em um sistema que recebe o nome de “Projeto Auxiliado por Computador” (*Computer Aided Design* - CAD).

Apesar de o uso de modelos digitais ter oferecido um grande impulso para a elaboração do projeto, trouxe também problemas na documentação e transferência das informações para a execução do produto. Quando do início do uso de modelos digitais a documentação do modelo do produto era feita através de desenhos em duas dimensões, elaborados de acordo com normas consistentes, sendo que as primeiras utilizações de modelos digitais também fizeram uso desta técnica para a documentação e transferência de informação.

Para que os benefícios da utilização de modelos digitais em sistemas computacionais sejam aproveitados no todo, faz-se necessária a completa

utilização do modelo digital em todas as fases do desenvolvimento do produto. Um exemplo pode ajudar a esclarecer esta necessidade: se o projetista deseja que um furo na peça tenha uma tolerância específica, ou um dado acabamento, esta informação precisa estar presente no modelo digital criado e persistir até a execução da peça. Sem o uso da tecnologia digital, isto era feito através de uma simbologia adequada no desenho. No modelo digital, de outra forma, mas com o mesmo objetivo, é preciso que esta informação seja armazenada e recuperada.

Com vistas a atingir o objetivo de fazer com que um modelo digital elaborado em um sistema computacional possa ser utilizado corretamente, com o aproveitamento de todas as suas características, desde a criação até a execução do produto, são utilizadas várias tecnologias.

Neste trabalho, serão abordadas quatro tecnologias que, em conjunto, pretendem atingir a meta da integração digital do ciclo de desenvolvimento do produto.

- Projeto baseado em *features* ;
- Paradigma da Orientação a Objetos;
- Linguagem de modelagem unificada - UML;
- STEP - Normas para a troca de dados do modelo do produto.

A tecnologia *form features* teve como objetivo inicial promover a integração entre projeto e manufatura [1]. O termo *form features* é utilizado, em uma visão bastante simplista, como a agregação da informação geométrica com mais informações necessárias a um determinado processo; desta forma, pode-se ter *design features, machining features*, *manufacturing features*, etc.

Nos primeiros trabalhos sobre a tecnologia *form features*, o processo consistia em reconhecer as *form features* possíveis de serem manufaturadas em um modelo digital já pronto. Outro enfoque desta tecnologia é a criação do projeto através de *form features,* no qual a criação do modelo digital é feita com a adição ou subtração de *form features*, tentando assim, reproduzir no modelo a execução do produto.

Este último enfoque levou a resultados positivos na integração ao longo do ciclo de desenvolvimento do produto, pois é possível, no momento do projeto, analisar

várias condições de manufatura e validar a intenção do projetista, dentro de restrições previamente estabelecidas.

O paradigma da orientação a objetos é uma técnica de modelar sistemas reais, criado com o propósito específico de geração de programas de computador [2]. Apesar de aparecer inicialmente em 1967, na linguagem de programação Simula, somente nos últimos 15 anos, com o aparecimento de processadores mais velozes e de maior disponibilidade de memória, que esta técnica de modelagem passou a ser utilizada [3].

Com o paradigma da orientação a objetos é possível modelar sistemas com uma complexidade bastante ampla, já que os sistemas são subdivididos em subsistemas mais simples que podem ser reutilizados.

A elaboração de um programa de computador, dentro do paradigma da orientação a objetos, provê a criação de um programa coeso, bem documentado e de fácil manutenção.

A linguagem de modelagem unificada (*Unified Modeling Language* - UML) foi criada em 1994, para a modelagem de sistemas dentro do paradigma de orientação a objetos [4]. Esta linguagem é composta de uma série de símbolos padronizados que permitem a completa descrição de um sistema real em um modelo orientado a objetos. Com a utilização desta linguagem para a criação do modelo tem-se a imediata vantagem da geração automática de uma documentação para o sistema e também a facilidade da verificação da validade do modelo durante a criação.

A utilização da linguagem de modelagem unificada também permite a utilização de sistemas de engenharia de software auxiliada por computador (CASE - *Computer Aided Software Engineering*), com geração automática do código em linguagem de programação.

A norma ISO 10303, Representação e troca de dados do produto, também conhecida como STEP (*Standard for the Exchange of Product Model Data*), foi criada com o objetivo de integrar a informação do produto durante seu ciclo de vida [5].

Com as normas ISO 10303 - STEP é possível ultrapassar a barreira imposta pela limitação de uma comunicação livre entre os vários atores do desenvolvimento do produto, por ser um modelo neutro.

Um dos produtos das normas STEP é um arquivo físico, no formato texto, que contém toda a informação do produto em uma forma definida. Este arquivo pode ser lido e interpretado corretamente, sem perda de informação por qualquer software que utilize as normas STEP.

Esta dissertação trata da combinação das mais modernas ferramentas tecnológicas disponíveis para atingir o objetivo de integrar o desenvolvimento do produto. A proposta é implementar contribuições em um modelador baseado em *features* para possibilitar que os dados do modelo fiquem armazenados de acordo com as normas STEP. A implementação destas contribuições será elaborada dentro do paradigma da orientação a objetos com o uso da linguagem de modelagem unificada.

1.1 Projeto *FESTEVAL*

Apesar de alguns sistemas CAD existentes já terem implementado algumas *form features* em sua interface com o usuário, pode-se considerar que os sistemas CAD disponíveis comercialmente são baseados em um modelo puramente geométrico. A implementação das *form features* na interface melhora a relação com o usuário durante o processo de projeto, mas, mesmo neste caso, estas *form features* são estruturadas em um modelo geométrico como faces, arestas, vértices, etc.

Como conseqüência, não é possível integrar digitalmente estes sistemas com um sistema CAPP. Também é necessária a interferência humana para interpretar a informação geométrica e para traduzi-la em termos de *manufacturing features* (*holes*, *pockets*, *slots*, etc.), que podem ser finalmente processados pelo CAPP, gerando um plano de processo como resultado.

A relação entre projeto e planejamento ainda permanece unidirecional devido à falta de uma semântica de projeto que possa ser entendida por outros sistemas. A linguagem utilizada é o desenho 2D em papel ou um modelo geométrico sem nenhuma informação tecnológica.

O projeto *FESTEVAL* dá ênfase à integração de todos os sistemas digitais envolvidos no desenvolvimento da cadeia de processo. Para que isto seja alcançado, o ponto inicial é a definição de uma estrutura com semântica comum para modelar as peças, que possa ser interpretada pelos sistemas subseqüentes. Esta semântica deve ser baseada no conceito de *form features*.

O projeto de pesquisa *FESTEVAL* proporcionou, além dessa dissertação, as seguintes publicações e relatórios técnicos:

LEIBRECHT, S: Development of a design environment with STEP processor for a feature base 3D-CAD System. Universidade Metodista de Piracicaba. Diploma Thesis - 2000.

SCHÜTZER, K.; GLOCCKNER CH.: Integration of Machine Operator know-How in a Feature Based Environment - CAD/CAPP/CAM/CNC. In: Proceedings of First Internacional Workshop on Intelligent Manufacturing Systems, Lausanne, Abr. 1998, pp. 67-84.

SCHÜTZER, K.; GLOCKNER, CH.; CLAASSEN, E.: Support for the Development Process chain by Manufacturing Features. In: Anais do 3º Seminário Internacional de Alta Tecnologia - Desenvolvimento Distribuído do Produto, Santa Bárbara d'Oeste, Out. 1998, pp. 133-145.

SCHÜTZER, K.; FOLCO, J.C.; GARDINI, N.: Modelador Baseado em "Manufacturing Feature" para Validação de Dados de Manufatura. Ciência e Tecnologia. 7 (1999) 13, pp. 81-88. SCHÜTZER, K.; SOUZA, N.L. DE: O Desafio de Integrar as Linguagens CAD/CAM no Setor Automotivo. Máquinas e Metais. 35 (l999) 404, pp. 74-83.

SCHÜTZER, K.; GLOCKNER, CH.; BATOCCHIO, A.: Implementação e Testes de um Ambiente Integrado de Projeto Baseado em "Manufacturing Features". In: Anais do COBEM' 99 - XV Congresso Brasileiro de Engenharia Mecânica, Águas de Lindóia, 1999. CD-ROM.

SILVA, N.A.; SCHÜTZER, K.; GLOCKNER, CH.; BATOCCHIO, A.: Integração de Sistemas CAD e CAPP Utilizando Tecnologia de Grupo e Modelamento Feature. In: Anais do COBEM' 99 - XV Congresso Brasileiro de Engenharia Mecânica, Águas de Lindóia, 1999. CD-ROM.

SCHÜTZER, K.; GARDINI, N.; FOLCO, J.C.: Modelador Baseado em Manufacturing Features Integrando Projeto, Processo e Fabricação. In: Anais do XIX Encontro Nacional de Engenharia de Produção, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 1999. CD-ROM.

SCHÜTZER, K.; FOLCO, J. C.: Modelador de Sólidos Integrado à Manufatura. In: Anais do III Encontro de Mestrandos em Engenharia, Santa Bárbara d'Oeste, 1999, pp. 35-41.

SCHÜTZER, K.; GARDINI, N.: Modelador Baseado em Manufacturing Features para Validação de Dados de Manufatura. In: Anais do III Encontro de Mestrandos em Engenharia, Santa Bárbara d'Oeste, 1999, pp. 28-34. 2000

CLAASSEN, E.; GYNDENFELDT, C. VON; SCHÜTZER, K.: Support of the CAD/CAM - Process Chain by Manufacturing Features. In: Anais do 5º Seminário Internacional de Alta Tecnologia - Inovações Tecnológicas no Desenvolvimento do Produto, Piracicaba, Out. 2000, pp. 111-132.

MOURA, A. A. A.; SCHÜTZER, K.: Utilização de Ferramentas para Implementação das Normas ISO 10303 - STEP. In: Anais do V Encontro de Mestrandos e I Encontro de Doutorandos em Engenharia, Santa Bárbara d'Oeste, 2001, pp. 113-122.

SCHÜTZER, K.; CLAASSEN, E.; GYLDENFELDT, C. VON: Support of the Product Development Chain by Manufacturing Features. In: Proceeding of the 7th Internacional Scientific Conference on Production Engineering - CIM 2001, Zagreb: Croatian Association of Production Engineering, 2001, pp. 29-40.

SCHÜTZER, K.; MOURA A. A. A.: Implementação das Normas 10303 - STEP em um Ambiente de Desenvolvimento Integrado do Produto. In: Anais do XXI Encontro Nacional de Engenharia de Produção, Salvador, 2001.

MOURA, A. A. A.; HENRIQUES, J. R.; SCHÜTZER, K.: Desenvolvimento Integrado do Produto Utilizando as Normas STEP - ISO 10303. In: Anais do Congresso de Pós-Graduação da UFSCar, São Carlos, 2001.

SCHÜTZER, K. et al. Improved software tools of the *feature* modeller software tools for recognition of most geometrical and technological interdependencies. SCPM-Dvd-14AC. Technical Report INCO-DC Project 96.2161. 2000

SCHÜTZER, K. et al. Requirements of *features* specification of the *feature*-based library. SCPM-Dvd-02. Technical Report INCO-DC Project 96.2161. 1998.

SCHÜTZER, K.; GARDINI, N.; FOLCO, J. Basic software tool of the *feature* modeller for the demonstrator. SCPM-Dvd-05. Technical Report INCO-DC Project 96.2161. 1998.

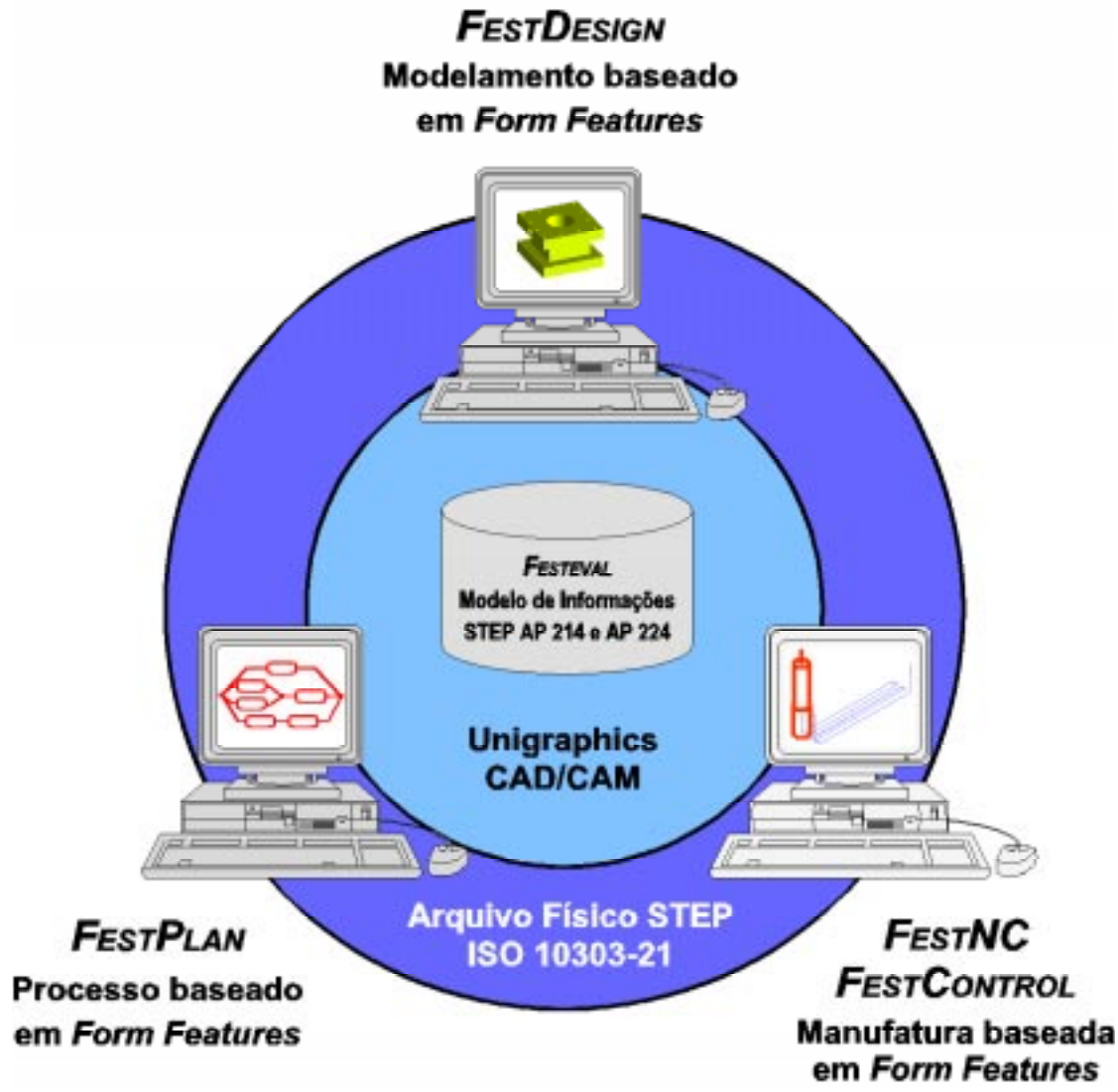


Figura 1.1: Módulos de desenvolvimento do projeto Festeval

O projeto *FESTEVAL (*Figura 1.1*)* foi desenvolvido com a atuação integrada de seis parceiros:

- Institut für Produktionsmanagement, Technologie und Werkzeugmaschinen - PTW/TUD

- Institut für Datenverarbeitung in der Konstruktion - DiK/TUD
- Laboratório de Sistemas Computacionais para Projeto e Manufatura - SCPM/UNIMEP
- Kade-Tech S.A.
- Indústrias Romi S.A.
- ProSTEP AG

Cada parceiro é responsável pelo desenvolvimento de uma parte do projeto, sendo a arquitetura do protótipo do projeto *FESTEVAL* apresentada na Figura 1.1.

O SCPM é o responsável pelo desenvolvimento do módulo de modelagem baseado em *features*, o *FESTDESIGN*. Este módulo auxilia os projetistas no processo de construção, gerando uma representação semântica, tecnológica e geométrica da peça, baseada em *form features*. O desenvolvimento do modelador é o objeto de estudo do trabalho aqui apresentado.

Os outros módulos do projeto são:

- *FESTPLAN,* que gera o plano de processo para a fabricação da peça disponível no modelo, desenvolvido pelo Kade-Tech.
- *FESTNC*, que auxilia a geração da trajetória NC para cada operação de usinagem e o *FESTCONTROL*, que controla o processo de usinagem NC na máquina ferramenta, desenvolvidos pelo PTW.

A integração dos módulos é feita através de um modelo de dados baseado nas normas STEP, desenvolvido pelo DiK.

As Indústrias Romi S.A. forneceram as informações sobre projeto e fabricação necessárias para a implementação dos módulos, e também avaliaram os protótipos implementados.

Como apoio para a modelagem em STEP e avaliação do modelo apresentado foram utilizados os serviços da ProSTEP.

Para a implementação do modelador baseado em *features*, o *FESTDESIGN*, foi necessário, primeiramente, escolher o sistema CAD que será utilizado, de acordo com seus métodos de construção e interface de programação. Depois disso, a interface com o usuário foi estudada e implementada garantindo sua eficiência e facilidade de acesso às funções implementadas. Por último, foram implementadas as funções para se obter as características e funcionalidades necessárias do protótipo, garantindo a construção e o armazenamento de produtos, seguindo o conceito de modelagem baseado em *features*. Para o encaminhamento deste trabalho foi feito um estudo do estado atual do desenvolvimento do produto que é apresentado no capítulo 2.

1.2 Conteúdo deste trabalho

Capítulo 1: Introdução

Descrição do projeto no qual está inserido este trabalho.

Capítulo 2: Desenvolvimento do produto - Estado da Arte

Descrição dos principais estudos sobre os temas utilizados no desenvolvimento deste trabalho.

Capítulo 3: Objetivos e Metodologias

Descrição dos objetivos deste trabalho e das metodologias utilizadas.

Capítulo 4: Desenvolvimento Teórico

Neste capítulo é tratada a teoria envolvida no desenvolvimento deste trabalho

Capítulo 5: Implementação

Descrição de modo como foi implementado o software do protótipo apresentado.

Capítulo 6: Conclusões

Capítulo 7: Referências Bibliográficas

2 Desenvolvimento do Produto - Estado da arte

No desenvolvimento do produto são utilizadas várias tecnologias. Nesta dissertação, serão abordadas as tecnologias diretamente envolvidas no desenvolvimento da interface STEP - ISO 10303, ou seja, o projeto baseado em *form features*; as próprias normas STEP e o desenvolvimento de software dentro do paradigma da orientação a objetos. O estudo destas várias tecnologias permitiu o delineamento do objetivo deste trabalho que é apresentado no capítulo seguinte.

2.1 Modelo do Produto

Em geral, um modelo é artificialmente construído para facilitar o estudo de um caso real. Com o intuito de possibilitar a observação, são criados maquetes, modelos moleculares ou modelos matemáticos [6].

O desenvolvimento de um produto de manufatura, iniciado pelo projeto auxiliado por computador, passa pela construção de um modelo digital. Este modelo utiliza as características gerais de qualquer modelagem, ou seja, é um dispositivo teórico que tem equivalência dentro de um escopo com o objeto real, tendo, portanto, *"um conjunto de hipóteses, suposto completo, relativamente ao domínio formado, e cuja coerência e desenvolvimento dedutivo, estão garantidos por uma codificação geralmente matemática."* [7]

A utilização dos recursos computacionais na criação de um modelo digital permite que seja criado, virtualmente, um modelo muito próximo do modelo real. Com os recursos hoje disponíveis, o modelo do produto tem todas as características de um modelo teórico e também, as características de um modelo material, sendo que estas últimas são características virtuais.

O modelo material é sempre transitório [8], como assim é o modelo digital criado através do projeto auxiliado por computador, já que este é apenas uma etapa para a construção física do produto. No entanto, como este modelo agrega também o modelo teórico que é perene, ocorre, na criação do modelo do produto, a justaposição dinâmica dos aspectos digitais, analógicos, materiais e teóricos do modelo [8].

No domínio específico do modelo do produto, os aspectos teóricos podem ficar isolados dos aspectos materiais do modelo. Os aspectos teóricos do modelo ficam restritos às características gerais da criação do modelo e os aspectos materiais, mesmo que virtuais, pertencem ao domínio específico do produto a ser modelado, instância do modelo teórico.

A construção de qualquer modelo obedece a regras de lógica e pressupõe uma sintaxe que permita, de maneira inequívoca, a descrição dos objetos pertencentes ao modelo e às suas propriedades, e também a de propriedades, que não estão vinculadas a nenhum objeto específico e sim à generalidade do modelo [7].

Com esta sintaxe, é possível formar regras para a construção do modelo. Estas primeiras regras constituem expressões que relacionam uma propriedade a um objeto e são chamadas de fórmulas elementares.

A fase seguinte à criação das fórmulas elementares constitui-se na introdução de operadores lógicos ou conectores entre as fórmulas já construídas. Inicialmente os dois conectores lógico-matemáticos de negação e implicação são suficientes.

Com o estabelecimento das fórmulas elementares e das conexões entre estas fórmulas, resta dar uma forma dedutiva a este feixe de hipóteses para a produção de outras fórmulas também válidas, usando, para isto, as regras de generalização e de separação.

Construído o modelo como descrito acima, cabe agora fixar o domínio de objetos fundados em uma correspondência com os símbolos do sistema, ou seja, aplicar, ao modelo teórico, uma estrutura composta de:

- Um conjunto não vazio, chamado de domínio ou universo (um objeto do modelo obrigatoriamente pertence a este conjunto);
- Uma família de subconjuntos pertencentes ao domínio estabelecido;
- A propriedade de “verdadeiro” ou “falso” a ser aplicada aos objetos do modelo.

Com esta estrutura criada e aplicada ao modelo, é possível efetuar a validade do modelo criado com o mundo real.

A descrição filosófica da construção de um modelo está voltada essencialmente para o estudo de um sistema real. Cria-se um modelo matemático para o estudo de estruturas metálicas, ou para o comportamento biológico de uma lagoa de tratamento de esgotos. O estudo do modelo do produto parte da idéia de um conjunto de hipóteses teóricas que são pertinentes a todos os produtos a serem criados. No entanto, todos os aspectos filosóficos do modelo são válidos para o modelo do produto. O projeto auxiliado por computador cria um modelo virtual, para depois ser criado o produto físico, real.

O objetivo final do projeto auxiliado por computador é a criação de um modelo do produto que contemple tanto aspectos teóricos quanto os materiais. Os aspectos teóricos validam a execução física do produto.

2.2 As normas STEP

Várias foram as técnicas desenvolvidas para superar as barreiras existentes na troca de dados entre os sistemas CAD.

A opção mais simples seria utilizar o mesmo sistema CAD durante todo o projeto, porém existem várias dificuldades com esta opção, desde exigências funcionais, como organizacionais. Em um ambiente internacional de parceria, é, geralmente, pouco prático insistir que mesmo os sócios principais usem o mesmo sistema e versão. Embora ainda exista uma pressão imposta pelas grandes empresas para que os fornecedores utilizem o mesmo sistema CAD, esta opção não se apresentou como sendo uma boa solução.

A segunda opção surgiu através do reconhecimento da necessidade de se ter múltiplos sistemas CAD e para isto houve a necessidade de construir tradutores entre estes vários sistemas. Com esta opção, não há dificuldades, se forem somente dois sistemas. Mas, para três ou mais sistemas, o número total dos tradutores requeridos, dado pela fórmula N(N-1) em que N é o número dos sistemas, já é significativo; e no caso de atualização de somente um sistema existe a necessidade de atualização de 2(N-1) tradutores, veja Figura 2.1. O trabalho de reintegração do sistema atrasará a execução da produção até que os sistemas sejam atualizados e testados, isso sem levar em conta o gerenciamento dos dados para as múltiplas versões de tradutores.

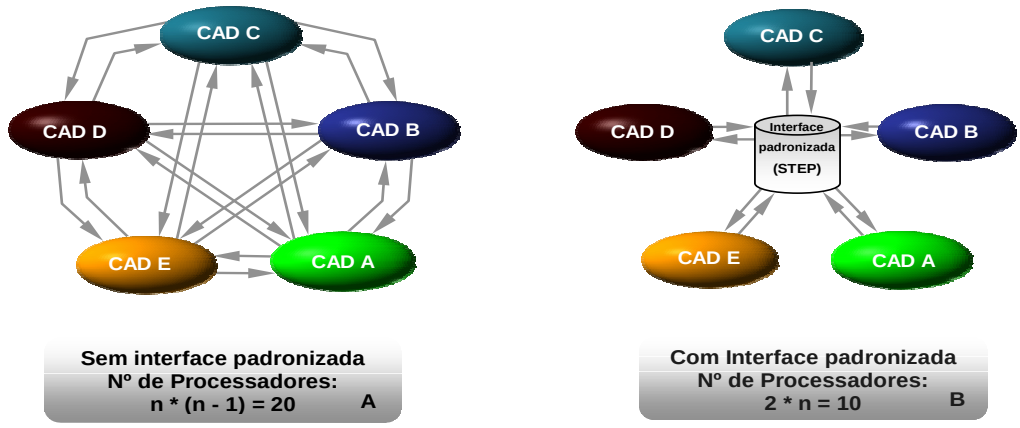


Figura 2.1: Troca de informações sem e com interface padronizada [9]

STEP (*Standard for the Exchange of Product Model Data*) é um conjunto de normas que dão apoio ao desenvolvimento do produto ao longo de todo o ciclo de vida, de forma que as informações permaneçam integradas e consistentes e não percam sua integridade [10].

STEP é uma norma ISO (*International Organization for Standartization*), e o seu código é ISO 10303. Ela está sendo desenvolvida pelo comitê internacional ISO TC 184/SC4 (Dados industriais) para abranger todos os dados de engenharia e não somente elementos gráficos como é feito pela IGES [10].

Os dados técnicos industriais, com o uso das normas STEP, ficam disponibilizados em um formato neutro, o que representa uma série de vantagens. Destas vantagens, podem ser destacadas a facilidade na troca de dados entre os diversos sistemas CAD e a possibilidade de compartilhamento das informações entre as empresas e setores internos.

As normas STEP estão construídas em uma linguagem bem estruturada que descreve formalmente a estrutura do produto, tornado assim um modelo confiável.

A implementação de uma empresa realmente globalizada depende da utilização de um modelo e uma linguagem comum que integrem todos os dados do produto ao longo de seu ciclo de vida.

Estudos feitos na Alemanha em 1998 mostram que somente a indústria automobilística e seus 900 fornecedores utilizavam aproximadamente 110 diferentes tipos de sistemas CAD [11]. Com um ambiente tão diversificado de

sistemas e com interfaces ineficientes de troca de dados, existe a geração de um altíssimo custo de retrabalho manual dos dados transferidos. Um dos caminhos para melhorar esta troca de dados é através do uso de PDT (Product Data Technology) e de uma interface para troca de dados, como a STEP (*Standard for the Exchange of Product Model Data*) [9].

As normas STEP dão o apoio necessário ao desenvolvimento do produto ao longo de toda sua cadeia de desenvolvimento, de forma que as informações permaneçam integradas e consistentes e que não percam sua integridade. Com isto, existe um aumento na eficiência dos sistemas de troca de informações entre os diversos setores da empresa, entre fabricantes e fornecedores e conseqüentemente reduzirá os custos envolvidos nas trocas de informações [1].

2.3 O projeto baseado em *FEATURES*

O objetivo de um projeto baseado em *form features* é obter um modelo que seja um conjunto de *features* sem redundância e suficiente para criar a peça real [12].

O uso da tecnologia de *form features,* no projeto, reduz o tempo necessário com retrabalho no projeto, reduzindo, conseqüentemente, o tempo entre a concepção e a comercialização (*time-to-market*) e o custo do produto. Além disso, acelera o processo da reutilização do projeto de novos produtos [12].

São várias as definições de *form features*, cada qual voltada para uma aplicação específica. De qualquer forma, independente de uma definição específica, a utilização da tecnologia de modelar com *form features* mostrou uma significativa melhoria na qualidade e no tempo de elaboração do projeto.

Uma definição genérica de feature pode ser dada por “um conjunto de informações de alto nível, definindo um grupo de características ou conceitos com um sentido semântico para uma etapa específica do ciclo de vida do produto” [13].

Desta definição podem ser destacados quatro pontos principais, sobre os quais se apóia a definição:

a) Uma *feature* tem informações de alto nível com sentido semântico;

b) Características físicas, não físicas e conceitos podem ser definidos;

c) É possível a definição e o uso de *features* durante o projeto;

d) Uma *feature* está ligada a uma etapa específica do ciclo de vida do produto.

Destes pontos, pode ser observada a estreita ligação da utilização de *features* com as informações do projeto. Em qualquer das definições é verificada a ligação desta com informações.

As tarefas de um projeto baseado em *form features* são: a escolha da *form feature*, a localização e a especificação da *form feature*, a especificação de tolerâncias e atributos tecnológicos e as especificações para outras etapas do ciclo de vida do produto [14], conforme pode ser visto na Figura 2.2.

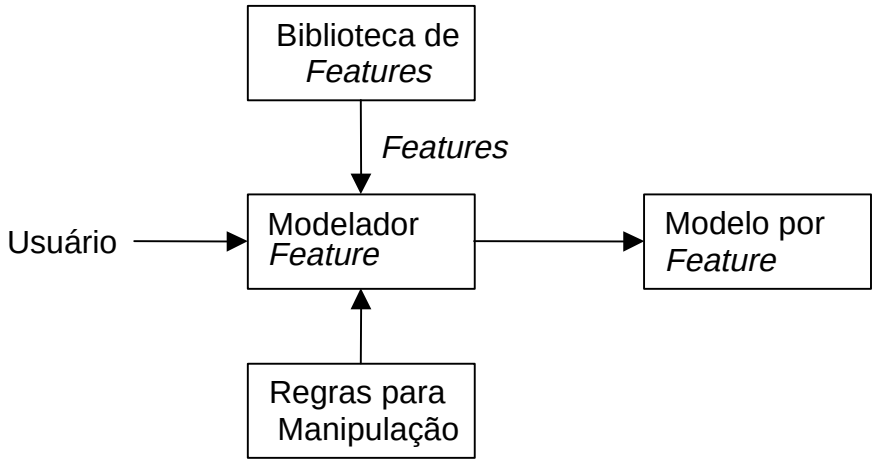


Figura 2.2: Esquema da criação de um modelo baseado em features

O objeto final do projeto, baseado em *features,* é o modelo criado pela agregação de um conjunto de *features* escolhidas do conjunto de classes disponíveis e manipuladas dentro de regras preestabelecidas. Existem, portanto, dois componentes principais para a modelagem baseado em *features*: as *features* disponíveis na biblioteca de classes *features,* e as regras para manipulação das *features*.

Em um sistema baseado em *features*, sua biblioteca é um conjunto parametrizado de informações composto de tipos ou classes de *features*, já que seu número é infinito mas estas podem ser categorizadas em um número finito de classes [15]. Dos tipos existentes na biblioteca podem ser criadas ou modificadas *features*

especialmente para o projeto [13]. Um modo de se criar esta biblioteca é utilizar o enfoque do paradigma da orientação a objetos e diagramas de classe da OMT [16].

O conjunto de classes de *features* deve ser organizado de forma taxionômica contendo as *features* usadas para um domínio específico [17].

As *features* instanciadas pela interação do usuário são inseridas no modelo através das definições dos vínculos geométricos e as inter-relações que são usadas para criar a geometria do produto, assim estabelecendo associações entre os objetos e o projeto [18].

Dentro desta definição, um modelo baseado em *features* que representa um produto (P) pode ser definido por um conjunto de *features* (Fi) e suas relações (δij):

$$P = \sum_{ij} \left(\delta_{ij}\right) \times \left(F_i\right)$$

A utilização da somatória das operações destaca que o modelo do produto não é somente um conjunto de *features*, mas sim a somatória do produto das suas inter-relações.

As relações entre as *features* definidas por $\left(\delta_{ij}\right)$ mostra que o modelo inclui relações entre uma *feature* $\left(F_i\right)$ com todas as outras *features* em P [17].

Um dos principais objetivos da modelagem baseada em *features* é a integração entre as etapas do ciclo de vida do produto, pois apesar dos sistemas CAD e CAPP já existirem há pelo menos duas décadas, a maioria das atividades de projeto e planejamento do processo é feita em separado. A distância entre o CAD e o CAPP é um dos maiores obstáculos para automação de processos [19].

A integração entre as etapas do ciclo de vida, uma das principais necessidades da engenharia simultânea, pode ser atendida através de uma representação adequada do produto, entretanto, uma dificuldade é não haver uma metodologia para o desenvolvimento do sistema e das operações, principalmente se a modelagem for feita com o único objetivo de criar o modelo geométrico, pois dificulta o compartilhamento dos resultados do domínio do projeto para outras áreas.

Uma solução possível é usar um modelo de produto que integre todas as áreas do conhecimento, porém a criação do modelo do produto, e do planejamento do processo são altamente dependentes de modelos do produto muito bem definidos, o que resulta em diferentes modelos de um mesmo produto, dando margem para vários conflitos devido à redundância de dados e perda de informações.

Dentro deste contexto, o planejamento do processo é que pode fazer o elo de ligação entre projeto e manufatura.

O uso, por si só, da tecnologia *features* não garante esta integração, pois a terminologia *feature* é usada para definir características específicas e estas características podem estar vinculadas somente a alguma etapa do ciclo de vida do produto.

Do ponto de vista do projetista uma *feature* pode ser definida pela sua função específica de projeto, enquanto que do ponto de vista da manufatura uma *feature* pode ilustrar um processo de usinagem. Dentro de uma visão mais ampla compreendendo o projeto baseado em *features,* uma *feature* pode ser usada para mostrar a associação entre o projeto e a manufatura [18].

Uma *feature* para projeto pode ser definida como uma representação pré-definida de uma unidade de projeto. É preciso ressaltar que uma *feature*, para o projeto, não especifica nenhum grau de abstração ou de dimensões. Esta definição do tipo de *feature* só contém uma função ou forma, ou seja, é uma definição puramente semântica.

A informação utilizada no início de um processo de projetar é diferente da informação disponível durante a fase preliminar do projeto, que é o processo de solução de problemas os quais envolvem uma série de atividades como busca de informações, análise, manipulação e estruturação da informação, geração de novas informações e avaliação do sistema de informações.

Numa operação de manufatura, uma ferramenta de corte é conduzida através de uma trajetória, enquanto o material é removido pelo movimento da ferramenta na peça. O volume resultante de uma operação de manufatura é uma *feature* de usinagem (*machining feature*) [20].

Do ponto de vista da manufatura, existe a idéia básica de que cada tipo de *feature* tem uma seqüência ordenada de estados a serem processados por operações [21].

Para o propósito de planejamento do processo, a qualidade de um produto, durante o projeto, é especifica em termos de qualidade superficial, tolerâncias, dimensionamento geométrico e tolerâncias geométricas [18].

O modelo criado com a tecnologia *features,* já pensado como um modelo único para todas as etapas do processo, permite que o processo de criação se dê em um ambiente amigável de projeto, pois facilita a interpretação geométrica relacionada a funções de projeto, avaliações de desempenho, processos de manufatura e outros processos de engenharia. Nos sistemas baseados em *features* há uma interação muito ampla entre aspectos geométricos e aspectos semânticos das *features* que permitem ao projetista associar atributos tecnológicos à geometria do modelo [22].

2.3.1 A criação do modelo baseado em *FEATURES*

A criação do modelo de um produto é trabalho do projetista e este modelo irá servir para todas as etapas seguintes do ciclo de vida do produto. Imediatamente após a criação do modelo do produto pelo projetista é feito o planejamento do processo, e até hoje o trabalho de projetista e planejadores tem sido feito com uma divisão bem definida em tempo e área. O resultado desta divisão é que, na ocorrência de erros de projeto, estes podem ser percebidos somente no planejamento do processo.

Estas dificuldades não foram vencidas pelo simples uso de sistemas CAx convencionais, principalmente porque os sistemas CAD convencionais não têm a capacidade de capturar aspectos não geométricos desejados pelo projetista como tolerâncias, inter-relacionamentos entre elementos, acabamentos superficiais, etc. Aspectos ainda mais abstratos como projetos conceituais, reusos, alternativas, funcionalidades são por si mais difíceis de serem representados.

O sistema baseado em *features* fornece a base para a integração das atividades de projeto, planejamento e manufatura, pois possibilita a agregação em uma

unidade das informações geométricas e tecnológicas para as operações seguintes [23].

Para a real implementação da engenharia simultânea é necessário ultrapassar a barreira da divisão entre projeto e planejamento de processo. O projeto do produto e o planejamento do processo devem, em conjunto com outras áreas, estarem integrados e, portanto, a troca de informações ao longo do ciclo de vida do produto é a chave para esta integração.

A intenção do projetista é um caráter abstrato do projeto que é vinculado ao modelo geométrico. Este caráter abstrato pode ser dividido em intenções geométricas e intenções funcionais.

Enquanto a intenção funcional está sempre implícita no modelo, a intenção geométrica precisa ser explicitada na ocorrência de uma interação entre volumes de duas *features*. Neste caso é preciso que o sistema indique exatamente a intenção do projetista [24], [25].

2.3.2 Interações entre *FEATURES*

Uma das chaves para o projeto baseado em *features* é o estudo das interações existentes entre elas. Deste estudo é feita a validação do modelo.

Quando duas ou mais *features* interagem, cada uma delas pode perder ou gerar informações vitais. Em sentido geral, o gerenciamento das interações entre *features* pode se revelar de extrema dificuldade por uma série de motivos, principalmente devido aos diferentes entendimentos sobre as *features*. Uma dificuldade adicional vem do uso de múltiplos conjuntos de *features*, cada conjunto adequado para uma aplicação específica.

O modo como as *features* interagem e a sua análise têm implicações em áreas críticas da pesquisa sobre *features*, inclusive para normalização e banco de dados [26].

A noção de interações entre *features* vem da influência que uma pode exercer sobre a outra. Essas interações podem ter vários efeitos no modelo, acarretando até na sua invalidação. Portanto, é necessária a verificação dos efeitos das interações entre *features*.

Uma primeira classificação das interações [26], as divide em três tipos:

- Interferência: Ocorre quando volumes canônicos de duas *features* têm uma interseção não nula, ou de outro modo, existe um volume compartilhado por duas *features*;

- Adjacência: É a interação entre *features* por uma adjacência entre duas *features*;

- Remota: Neste caso, mesmo com as duas *features* não compartilhando volumes ou adjacências, existe entre elas um vínculo qualquer, como por exemplo, um compartilhamento de algum atributo tecnológico.

Uma outra possível classificação das interações entre *features* e suas conseqüências é dada na Tabela 2.1, cujos exemplos estão na Figura 2.3.

	Tipo de interação	Descrição
A	Absorção	A feature não acrescenta nada ao modelo.
B	Desconexão	Uma parte do volume de uma feature fica desconectado do modelo.
C	Divisão	Os limites de uma feature são divididos em dois ou mais
D	Geométrica	Relações geométricas entre features
E	Oclusão	O volume de uma feature subtrativa fica fechado dentro do modelo, não há acesso á feature
F	Obstrução de face	Obstrução da face de uma feature subtrativa
G	Obstrução de volume	Obstrução do volume de uma feature subtrativa
H	Topológica	Relações topológicas entre as features
I	Transformação	Uma feature passa a ter características de outra classe ou tipo

Tabela 2.1: Interações entre features [27]

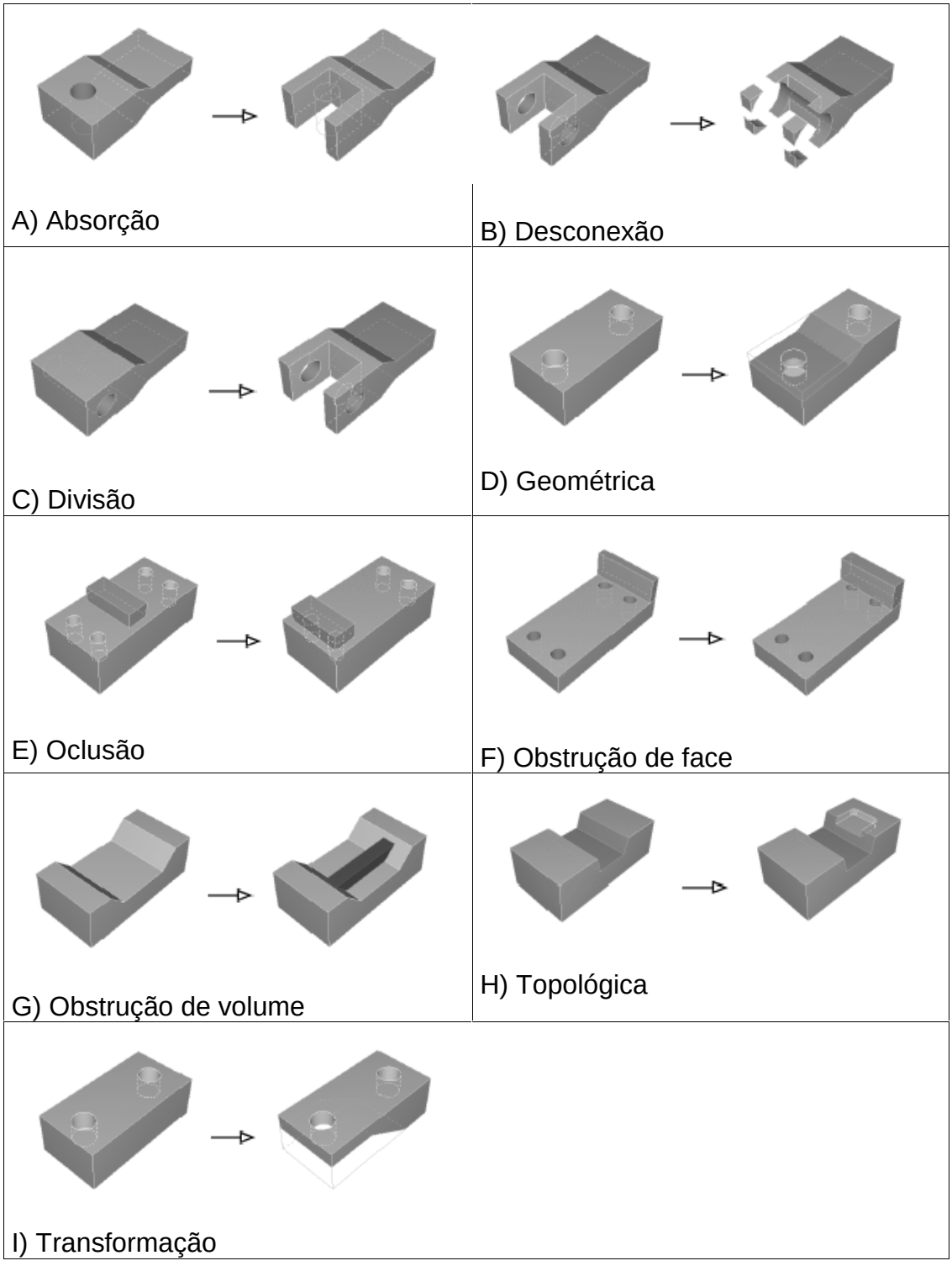


Figura 2.3: Exemplos de interações entre features [27].

No momento da criação do modelo, a instanciação de uma *feature* pede ao projetista uma série de parâmetros para sua inicialização. Alguns destes valores

são referenciados em outras *features* (e.g. faces), e servem para especificar como são feitos o posicionamento e anexação da nova *feature*.

Estas referências são persistentes, pois continuam válidas durante o tempo em que a *feature* estiver no modelo, criando uma interação entre as *features*, ocasionando uma clara interdependência no modelo.

Esta dependência entre duas *features* é unidirecional, isto é, é possível determinar a *feature* tutelanda da *feature* tutelada [28].

Se houver ferramentas para sugerir, aos projetistas, revisões que melhorem as características de manufatura de um projeto, isto causará uma redução do custo do produto e do tempo de produção.

Durante as modificações do projeto é preciso efetuar uma verificação contínua para garantir a validade do modelo.

O processo de validação pode se tornar muito extenso, se a validação for efetuada do ponto de vista do modelo; já a validação dos vínculos entre as *features* se mostra bastante eficaz e vantajosa.

Uma outra forma de especificar as interações entre *features* é a análise de seus vínculos. Os vínculos entre *features* oferecem uma maneira conveniente de especificar a intenção do projetista ao criar o modelo.

Os vínculos geométricos são os que definem relações tecnológicas entre as propriedades geométricas de uma *feature* ou entre *features;* como relações geométricas de paralelismo, perpendicularidade e distância.

Os vínculos topológicos podem ser de volume ou de face quando especificarem relações destas propriedades entre duas *features* [29]

Nesta análise, os vínculos geométricos correspondem às interações remotas, enquanto que os vínculos topológicos representam tanto interações de interferência e de adjacência.

Além disso, os vínculos armazenam a intenção do projetista. Eles podem ser relativos a uma *feature* em especial como tolerâncias dimensionais ou podem denotar inter-relações entre várias *features*.

Os vínculos entre as *features* podem ser definidos em vínculos de volume, de face ou de arestas

Os vínculos de volume definem um volume no modelo que tanto pode ser pertencente a uma ou outra *feature* e os vínculos de face determinam faces que podem pertencem a mais de uma *feature*, sendo que o mesmo ocorre com as arestas.

Segundo Chang [15], as relações entre *features* podem ser divididas em três categorias: “está-dentro”; “está-em” e “é-adjacente”. A relação “está-dentro” indica uma relação espacial entre uma *feature* positiva e uma *feature* negativa. A relação “está-em” pode ocorrer no caso de duas *features* negativa ou duas *features* positivas. As duas implicam em uma dependência geométrica entre duas *features*. A relação “é adjacente” denota uma adjacência geométrica entre duas *features*.

Para Dankwoort [30] as relações entre as *features* podem ser compostas de duas maneiras: topológicas ou geométricas.

São consideradas relações geométricas, as tolerâncias dimensionais ou tolerâncias de posição ou orientação da geometria das *features*.

As relações topológicas expressam as associações entre uma *feature* e as *features* adjacentes. São classicamente definidas em termos de “aberto em” ou “aberto para”, quando uma *feature* tem uma abertura para (ou em) outra *feature*; "intercepta com” quando duas *features* se interceptam. Estas relações são consideradas atributos de *features* de manufatura.

Desta forma, pode-se verificar que as pesquisas reduzem as interações sempre nos três tipos inicialmente propostos por Regli [26], seja pela análise das interações [15], [26], [30] ou pela análise dos vínculos [12],[28].

A análise de NORT [27] pode também ser reduzida aos três tipos descritos, se for introduzido o conceito de validação semântica, em que a mudança de classe não é válida para o projeto.

2.3.3 Validação semântica

Como a tecnologia *feature* inclui, no mesmo modelo, as informações geométricas e as tecnológicas, a simples verificação de satisfação das condições geométricas não é o bastante para a total aplicação da tecnologia [31]. A verificação das condições tecnológicas é feita através da validação semântica do modelo, também entendida como modelagem semântica.

Uma das características da modelagem semântica é que todo o processo de modelagem é uniformemente levado a termo com um conjunto de *features* e seus vínculos. Todas as ações de modelagem feitas pelo projetista constituem ações com *features,* pois todo o modelo é, efetivamente, baseado em *features* [28].

A validação semântica de modelos baseados em *features* não é completa sem o adequado gerenciamento das interações entre *features*, o que requer vários níveis de modelagem baseados em *features*, em particular a:

- Nível de especificação das classes de *features*;
- Nível da representação geométrica dos modelos;
- Nível de operação do modelador.

Em muitos sistemas de modelagem baseado em *features*, estas somente são utilizadas em nível de projeto; sendo assim, o modelo obtido tem somente a geometria. Estes sistemas constroem, essencialmente, modelos geométricos; em outros sistemas, são armazenadas também informações sobre as *features*, porém não há a verificação da consistência do significado de todas as *features* durante todo o processo de modelagem. Por exemplo, um furo cego pode se transformar em um furo passante pela abertura da face de fundo com a introdução de uma *feature*, sem que isso seja informado pelo sistema. Apesar de estar correto do ponto de vista geométrico, está incorreto do ponto de vista tecnológico; a semântica da *feature* não é mais válida.

De modo ideal, toda a validação tecnológica das *features*, i.e. da sua semântica, deve ser feita pelo sistema após cada modificação do modelo. Se alguma condição de validação não é verificada, deve ser notificada pelo sistema ao projetista. Esta solução garante que toda a intenção do projeto, uma vez obtida, é

mantida no modelo, dando um nível de significado realmente superior ao da modelagem geométrica.

A especificação das condições de validação nas classes das *features* pode ser dividida em duas categorias: geométrica e tecnológica. Para ambas, são utilizadas restrições para *features* as quais são definidas como membros das classes e são instanciadas automaticamente na criação da *feature* [32].

A manutenção da validade de um modelo baseado em *features* é o ponto central da modelagem semântica e consiste no processo de monitorar cada operação de modelagem, para assegurar que todas as *features* mantenham seus critérios de validação verdadeiros.

Dois princípios básicos são utilizados neste processo: 1) Uma operação de modelagem, para ser considerada válida, deve manter todas as *features* válidas para todos os critérios definidos; 2) Após uma operação inválida, o projetista deve ser informado para a correção da operação.

A Figura 2.4 apresenta um esquema geral para a execução de uma operação de modelagem com validação semântica [28], onde o processo de validação é reiniciado através de um laço de reação a cada modificação no modelo ou no caso da ocorrência de um erro no processo.

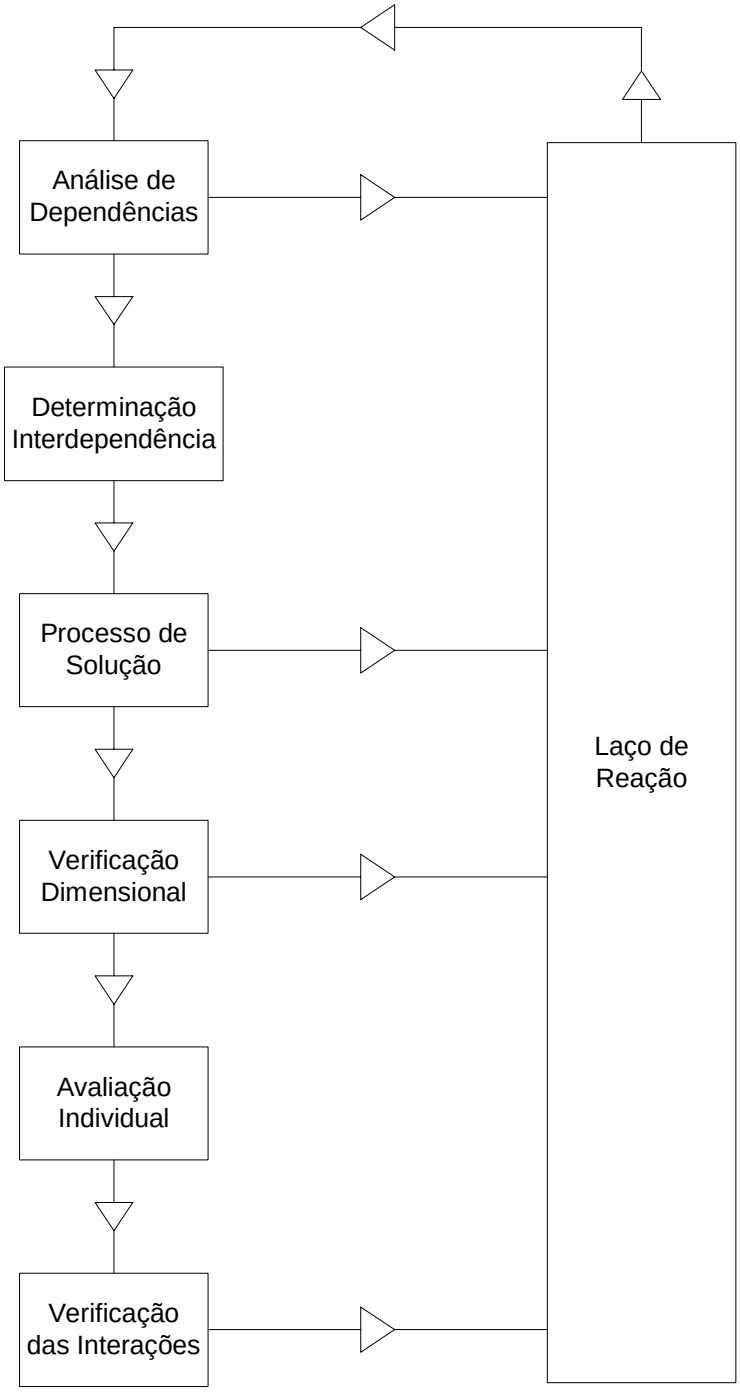


Figura 2.4: Esquema para processo de validação semântica [28].

2.3.4 Atributos Tecnológicos

A base da tecnologia de *features* é o agrupamento de informações geométricas com informações tecnológicas; enquanto aquelas são básicas para os sistemas

CAD, estas devem ser incluídas de modo especial. Talvez a mais importante informação tecnológica, a ser incluída no conceito de *feature,* seja o conjunto de tolerâncias.

Existem dois tipos de tolerâncias mecânicas: paramétricas e geométricas. Uma tolerância paramétrica define um limite ou um par de limites aplicados a algum parâmetro escalar de um modelo. Esta tolerância é representada por um valor nominal mais um limite superior e inferior, expresso dentro de uma norma.

Uma tolerância geométrica define restrições à geometria da *feature*, i.e. a tolerância define uma zona na qual o formato da *feature* deve estar. As *features* ficam em uma zona representada em planos, cilindros, cones, esferas e paralelepípedos. Há diversos tipos de tolerâncias que controlam vários aspectos de forma e localização, tolerâncias de forma definem o desvio permitido da forma ideal ou nominal; tolerância de orientação define o desvio da orientação; tolerâncias de localização definem o desvio de posição. Muitos pesquisadores relatam a dificuldade de especificar as tolerâncias de modo preciso.

Neste cenário, a interação entre o projeto e o conjunto de tolerâncias fica dentro de um esquema de interações.

2.4 A Integração Digital no Desenvolvimento do Produto

Os sistemas de tecnologia da informação têm o objetivo de integrar os processos comerciais pela oferta de canais de comunicação que permitam aos membros de uma equipe a troca de dados e informações pelas barreiras físicas e temporais. Esta promessa, no entanto, não é cumprida na totalidade. Os dados são armazenados e duplicados em tantos lugares diferentes que, às vezes, um dado “bom” se torna um dado “mau” mesmo quando alguém consegue obtê-lo, e este não é o maior obstáculo que os esforços para integração deve vencer. O maior problema é o fato de que os dados são diferentes para as etapas de criação e utilização, o que significa que os dados devem ser convertidos entre as aplicações [33].

Durante as duas últimas décadas, pesquisadores efetuaram vários esforços para modelar e integrar a informação requerida para o planejamento do processo em uma manufatura integrada por computador. Huang [34] desenvolveu um

componente de modelo baseado em *features* para CAPP somente do ponto de vista do projeto. Zhang [35] e Norrie [36] apresentaram um modelo para um ambiente de desenvolvimento integrado, mas muito geral para a informação necessária ao CAPP. Eversheim e Marczinski [37] geraram uma modelagem estrutural de processo de manufatura projetado especialmente para a preparação da manufatura. Apesar de todos estes esforços, não foi possível a representação do conhecimento inserido nos modelos digitais, por isso, a maioria dos sistemas CAPP não foi aplicada na indústria, onde o uso amigável, e também a efetiva cooperação e compartilhamento de informações são pré-requisitos. Como resultado, um modelo de informações eficiente é necessário para alimentar um sistema CAPP que tenha a capacidade de se integrar com outros sistemas em um ambiente de desenvolvimento integrado de modo conveniente e em última instância implementar todo o sentido do desenvolvimento integrado do produto em uma indústria.

O planejamento do processo tradicional é criado manualmente por um planejador humano, que lê o desenho de uma peça, e determina o processo apropriado para produzi-la; entretanto, há problemas acerca deste método manual, como alto custo e limitações, que foram reduzidos na década de 60 com o desenvolvimento de sistemas CAPP [11]. As pesquisas para estes sistemas resultaram em dois enfoques para o desenvolvimento de sistemas de planejamento do processo auxiliado por computados, chamados de sistemas de planejamento do processo variante e sistemas de planejamento do processo generativo [11].

Por serem os problemas de planejamento de processo extremamente dependentes de conhecimento, foram utilizadas técnicas de inteligência artificial e também sistemas especialistas para capturar, representar, organizar e utilizar o conhecimento em computadores de modo efetivo. A formulação e a solução dos problemas de planejamento do processo foram discutidas extensivamente na literatura.

Somente poucos resultados foram realmente utilizados na indústria, e muitos poucos acrescentaram melhorias substanciais às práticas de manufatura. Uma das principais razões é a falta de um método efetivo para representar toda a informação necessária para o CAPP, e também para unificar a informação de outros sistemas do ambiente de desenvolvimento integrado do produto.

De fato, a maioria dos modelos publicados se concentra em somente um aspecto da informação requerida pelo CAPP, ao invés de desenvolver um modelo de informação completo para englobar a informação necessária para executar as tarefas do CAPP e também se integrar aos outros sistemas de manufatura.

A modelagem, dentro do paradigma da orientação a objetos, é bem documentada para suas características de abstração, encapsulamento, herança, polimorfismo e vínculos dinâmicos. Estas características são essenciais no desenvolvimento de um sistema CAPP, porque dão consistência à modularidade, modificabilidade, reusabilidade, extensibilidade e robustez. As normas STEP foram especialmente projetadas e desenvolvidas para a integração entre vários sistemas em um ambiente de projeto integrado através da adoção de recursos genéricos, normas para implementação, etc.

Existem duas formas de se tentar prover a integração entre os sistemas CAx. Uma delas consiste em incluir, nas informações das *features,* toda a informação necessária para as outras etapas do processo. Uma das críticas a esta proposta é que uma *feature* específica para a fase de projeto tem limitações para os próximos passos [38].

Outra proposta concebe que uma *feature* de projeto é ideal para a construção do modelo geométrico, mas outras *features* podem suprir com mais consistência as informações [38].

2.5 Modelo de dados do produto

Em um passado recente, a forma essencial de modelagem consistia em representar o processo de projeto e formalizar a informação necessária.

Os primeiros modelos de dados baseados no conceito entidade-relacionamento foram introduzidos na década de 70 [39].

A idéia de um modelo de dados do produto é prover os meios necessários para a representação e armazenamento da informação sobre um produto desde o projeto até a manufatura do produto. Entretanto duas barreiras devem ser transpostas para atingir este objetivo. Devido aos diferentes objetivos de cada aplicação, diferentes pontos de vista dos dados de produto são necessários, e os atuais sistemas CAD, por utilizarem um modelo puramente geométrico, não são

capazes de fornecer toda a informação de uma peça necessária para todas atividades destas aplicações.

Há uma busca intensa em produzir banco de dados que possam organizar, de modo discreto, os dados de projeto, manufatura e planejamento do processo e também suas associações.

Um modo de fazer esta associação é a utilização da idéia de um modelo mestre [40], um repositório orientado a objetos, que provê mecanismos essenciais para a criação e validação de um modelo particular (veja Figura 2.5). Este modelo mestre pode ter vários clientes sendo que um deles é o sistema de CAD. O modelo mestre é o conjunto de todas as *features* possíveis, armazenadas dentro do paradigma da orientação a objetos. Deste modelo, são obtidas as *features* que irão compor o modelo final, de modo a servir vários clientes, incluindo aí o sistema CAD.

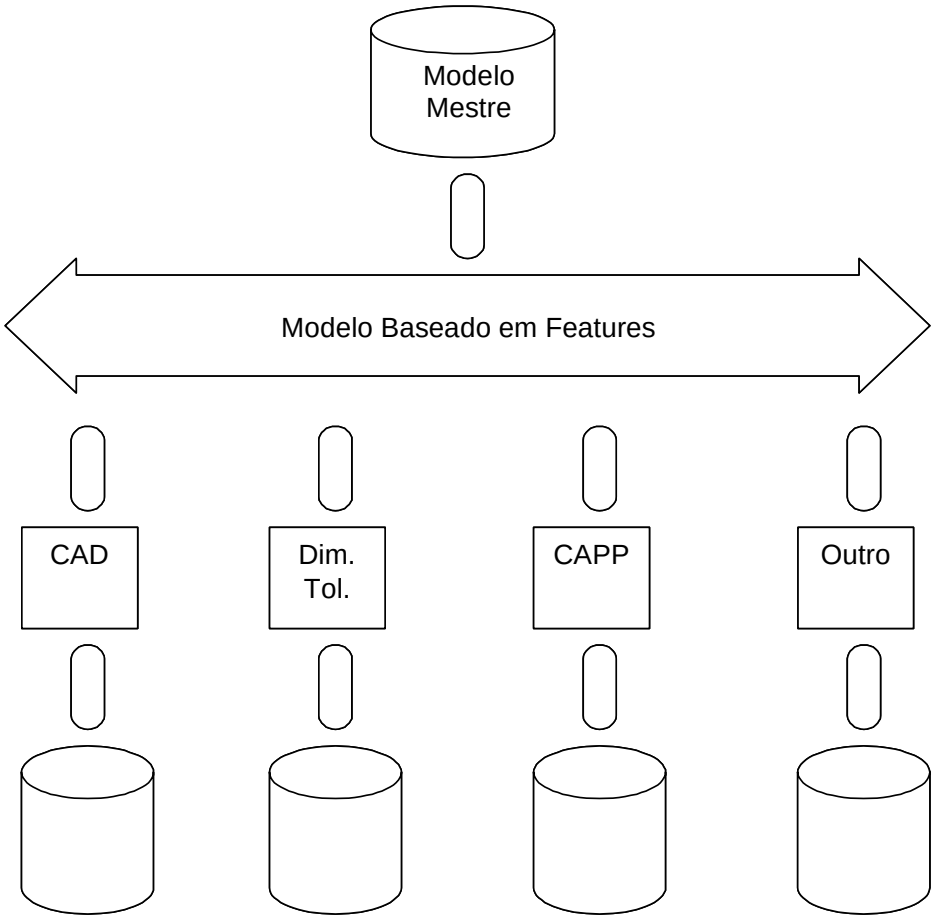


Figura 2.5: Biblioteca e modelo baseado em features [40]

O conjunto de dados do produto permite ao sistema CAD auxiliar o projetista durante o desenvolvimento do projeto do produto. Com a aplicação automática de procedimentos já conhecidos, é possível, das condições iniciais, criar soluções alternativas, fazendo a verificação em tempo de projeto quanto à compatibilidade das mudanças específicas no modelo do produto comparando sempre com as condições iniciais.

Dentro deste espectro, surgiu uma solução promissora: a criação de um método padronizado de criação do modelo do produto que sirva para todas as aplicações. Esta idéia está contida nas normas STEP - *STandard for the Exchange of Product model data* [5].

O objetivo desta norma é prover um mecanismo neutro de troca de dados capaz de descrever a definição do produto durante todo o ciclo de vida do produto, o que requer a definição e representação de uma grande variedade de informações. Um meio de se conseguir isto é a utilização de protocolos de aplicação (AP); cada um deles dá uma visão especial dos dados necessários para uma ou mais aplicações.

O protocolo de aplicação AP 224: *Mechanical Product Definition for Process Planning using Form Features [41]* define uma visão específica para o planejamento do processo. O sistema resultante, baseado neste protocolo de aplicação, pode conter representação tanto de peças rotacionais quanto de peças prismáticas, incluindo atributos geométricos e tecnológicos (i.e. tolerâncias, acabamentos, etc.).

Este modelo do produto também descreve uma peça em termos das suas propriedades e de seu material. Estas propriedades estão divididas em quatro grupos: processo; superfície; material e quantidade [42].

A importância da utilização de normas para a troca de dados do produto vem da vantagem de prover uma representação pública e estável e as normas têm um processo de alterações baseada em consenso. Formatos proprietários de troca de dados podem e são modificados de acordo com os interesses dos desenvolvedores. Além disso, a atividade da manufatura é global.

As normas STEP despertam interesse devido à redução do tempo para o lançamento de um produto no mercado, como resultado da melhoria na

comunicação e acesso às informações do produto [43], e os próprios dados de um sistema CAD podem ser melhorados com o uso da tecnologia STEP em um modelo único e com o uso da linguagem EXPRESS [44].

2.5.1 As normas STEP - ISO 10303

As primeiras normas com o propósito de resolver o problema da troca de dados foram: a norma IGES (Initial Graphics Exchange Specification), SET (Standard d'exchange et de Transfer) e VDA FS (Verband der Deutschen Automobilindustrie Flächenschnittstelle), que evoluíram em várias versões, conforme pode ser visto na Figura 2.6. Todas estas normas, no entanto, especificam apenas um formato neutro: os diferentes sistemas de CAx transferem seus bancos de dados ou partes e produzem um arquivo com um formato especificado. Os principais problemas com estas tecnologias, baseadas em formatos neutros, são a redundância de informações e interpretações ambíguas destas informações que produzem perdas no modelo de dados. Informações duplicadas ocasionam o armazenamento de dados inúteis e dificultam a manutenção do sistema, pois requerem o gerenciamento de dois conjuntos de dados (o nativo e o neutro) atualizados. A idéia de trocar o modelo de dados do produto ao invés dos dados do produto vem da experiência da norma IGES [5].

O objetivo é determinar quais são as classes necessárias e, nestas, quais atributos e métodos devem servir de apoio para os sistemas CAD para maximizar a reusabilidade do código e prover todos tipos de informação para representação e manipulação no contexto do sistema CAD. Os principais requisitos são [45]:

- A arquitetura deve permitir a integração das ferramentas para todas as áreas do projeto mecânico;
- A arquitetura deve ser independente da plataforma e da linguagem;
- As ferramentas de software devem compartilhar dados de visualização e do modelo;
- As ferramentas de software não devem necessitar de alterações em seus códigos-fontes;

- A arquitetura deve permitir resposta em tempo-real aos eventos entre ferramentas.

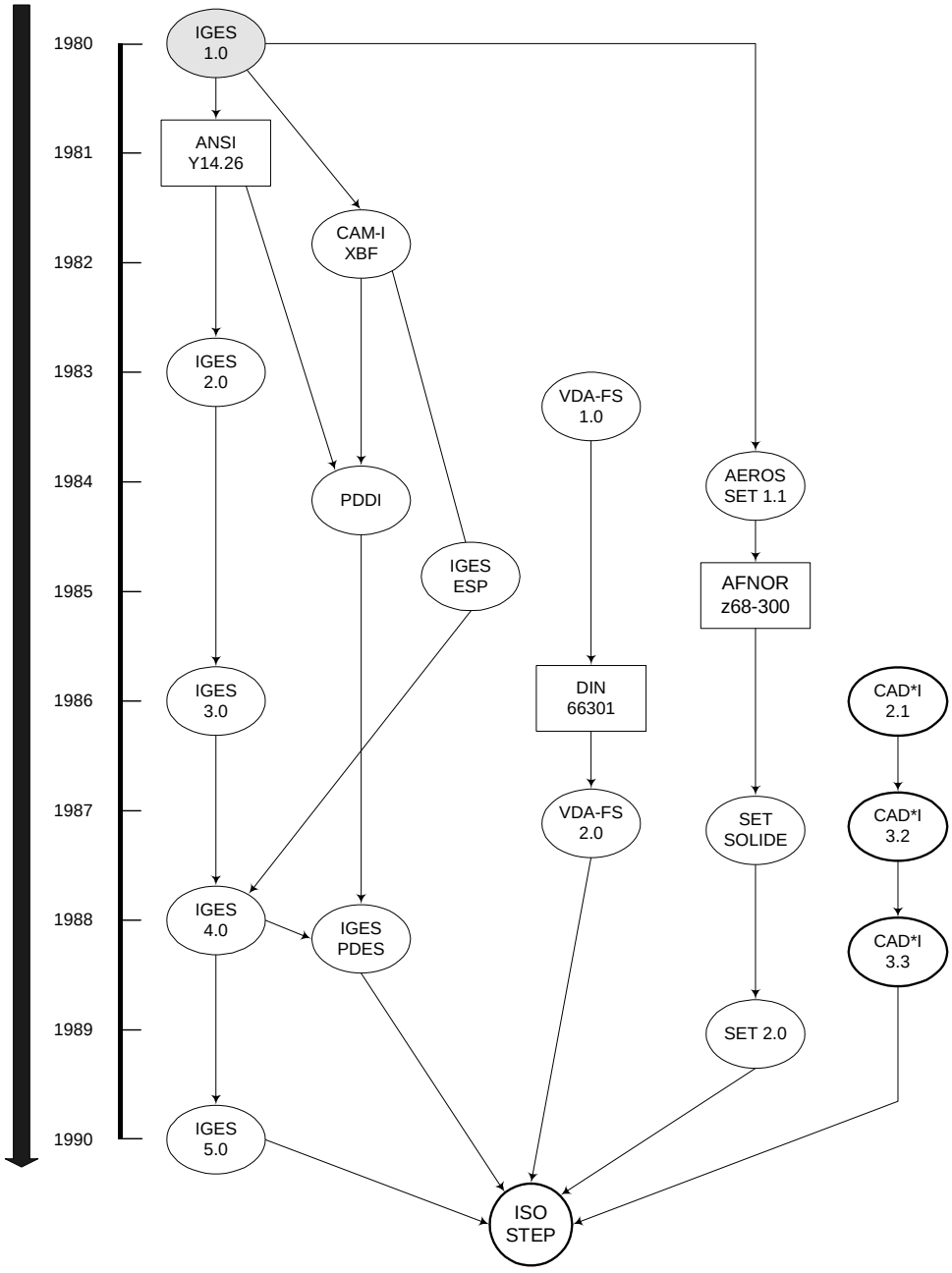


Figura 2.6: Evolução das normas para troca de dados do produto.

Uma das diferenças em relação as normas STEP é que sua concepção é está elaborada em um grau de abstração de forma a abranger qualquer informação

sobre o produto pode ser compartilhada e não somente os dados do sistema CAD [46].

2.5.2 Estrutura geral da norma

As normas STEP começaram a ser implementadas na reunião da ISO em Frankfurt, Alemanha, em 1989. A atual estrutura das normas STEP é conseqüência de um grande número de contribuições recebidas [47].

Com a meta de atingir os objetivos propostos, as normas STEP foram desenvolvidas em uma estrutura modular para garantir a flexibilidade necessária. Os vários módulos são documentados em partes separadas na Norma Internacional ISO 10303 [9].

O grupo de definições de dados do produto que podem ser suportados pela STEP é subdividido em um número de módulos independentes com interfaces claramente definidas. Estas interfaces foram definidas por pessoas especializadas e que atuam na área e depois integradas por um time central em uma representação consistente.

Os métodos e princípios utilizados para o desenvolvimento da STEP foram divididos em cinco categorias principais [48],[49]: métodos de descrição; metodologia de implementação; recursos para a integração da informação; metodologia de testes para conformidade e estrutura de trabalho; protocolos de aplicação e grupos associados de testes abstratos [48] (veja Figura 2.7).

A linguagem EXPRESS, definida na própria série de normas STEP, descreve todos os módulos.

Para a troca de dados entre sistemas CAD, as normas STEP providenciam um método de implementação para cujo objetivo final é um *arquivo físico* que é um arquivo em formato texto seqüencial [16],[11].

Como não é possível ter, em um único conjunto de dados, todas as aplicações possíveis e, além disso, conter um contexto concreto para os dados, as normas STEP utilizam Protocolos de Aplicação que descrevem a utilização da troca de dados para uma aplicação particular.

Como exemplo: o protocolo de aplicação 214 para a indústria automobilística, ou o protocolo de aplicação 224 para o planejamento do processo baseado em *features* de manufatura. Cada protocolo de aplicação possui a definição de uma área de negócio e as informações requeridas por esta área.

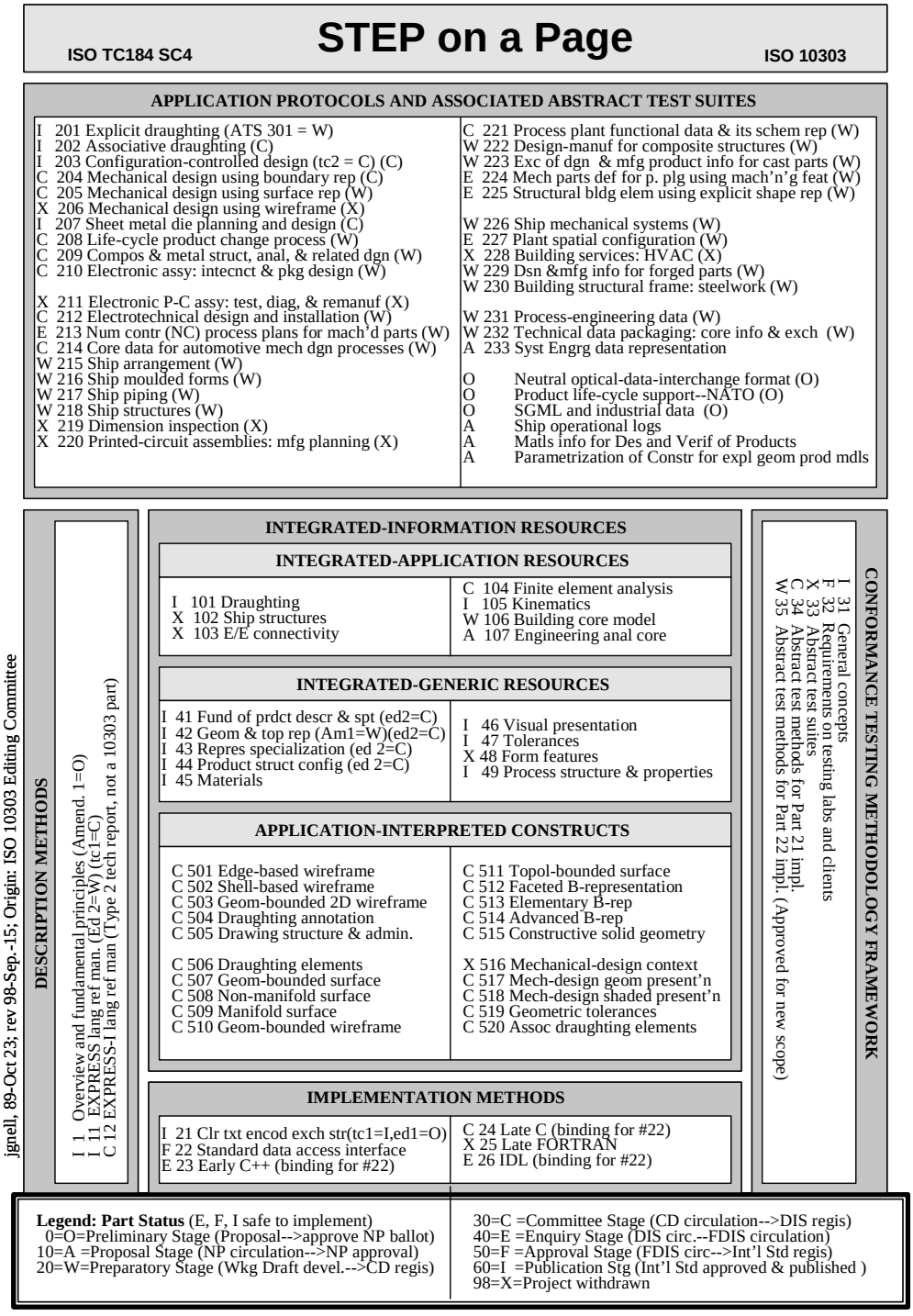


Figura 2.7: Estrutura geral das normas STEP [48]

2.5.3 Protocolos de Aplicação

Uma das características das normas STEP é a capacidade única de adequar e expandir-se de modo a atender aplicações específicas sem alterar a consistência central das normas. Esta característica é alcançada com a organização da norma em protocolos de aplicação (AP - Application Protocols), que são conjuntos de entidades escolhidas para um produto específico para um processo ou indústria. Por exemplo, existem protocolos específicos para a indústria automobilística, aeroespacial e naval, bem como para os projeto de estamparia. Cada protocolo de aplicação é um documento formal que descreve:

- Uma etapa do ciclo de vida do produto, AAM (*Application Activity Model*);
- As partes de informação do produto necessárias para estas atividades, ARM (*Application Reference Model*);
- Como utilizar os métodos construtores das classes do modelo, AIM (*Appplication Interpreted Model*).

O protocolo de aplicação AP 224 é a norma da série STEP para “definições do produto mecânico para planejamento do processo usando *features* de manufatura”. Este protocolo pode ser entendido, de modo geral, como uma biblioteca de *features* de manufatura parametrizadas, mas também dá definições das *features* de manufatura em termos de B-Rep e outros dados como exceções de projeto e requisitos de manufatura [50].

A biblioteca de *features* do AP 224 fornece métodos plenos para a descrição das *features* em detalhes [51].

2.6 Linguagem EXPRESS

A linguagem EXPRESS foi desenvolvida dentro da família STEP como uma norma ISO, para providenciar a especificação formal da representação e troca de dados do produto. É especialmente projetada para especificar o modelo de informações que representam um número de passos no ciclo de vida do produto. EXPRESS fornece a sintaxe e a semântica para representar a informação em um modo uniforme, preciso e compacto. A representação em EXPRESS é possível

de dois modos: o primeiro, como uma linguagem formal que usa uma notação léxica e uma sintaxe definida por uma gramática própria; o segundo, como uma representação gráfica (chamada EXPRESS-G), que fornece uma ilustração bastante compacta e rica e, acima de tudo, amigável.

EXPRESS é uma linguagem de descrição de dados orientada a objetos. Está estruturada em esquemas que representam o modelo do produto. Um esquema consiste em entidades, que são os principais objetos e tipos de dados na qual se apóiam as definições destas entidades. Dentro das entidades estão encapsulados atributos e vínculos, que restringem os valores dos atributos. Um esquema EXPRESS também tem declarações de FUNÇÕES, PROCEDIMENTOS e REGRAS que restringem uma ou mais entidades ou tipos de dados. A linguagem EXPRESS é definida na norma ISO 10303-11. A Figura 2.8 apresenta um exemplo de um esquema com as entidades: ponto e ponto cartesiano.

```
SCHEMA example_geometry;
TYPE lenght_measure = NUMBER;
END_TYPE;
ENTITY point;
SUPERTYPE OF (ONEOF (cartesian_point) )
END_ENTITY;
ENTITY cartesian_point;
SUBTYPE OF (point);
X_coordinate: length_measure;
y_coordinate: length_measure;
z_coordinate: OPTIONAL length_measure;
END_ENTITY;
END_SCHEMA;
```

Figura 2.8: Exemplo de uma entidade em EXPRESS [47]

A linguagem EXPRESS descreve um domínio de informação em termos de entidades que são puramente dados de descrição; não há, nesta linguagem, nenhuma função para acessar ou manipular dados [52].

Os conceitos de modelagem da linguagem EXPRESS podem ser separados em cinco categorias: esquema, entidade, atributo, relacionamento e restrições.

Um **esquema** define o escopo e o contexto de um conjunto de entidades. As entidades de um mesmo esquema compartilham definições e semânticas comuns dentro de um mesmo contexto.

Uma **entidade** é sempre declarada dentro de um esquema e corresponde a um objeto do mundo real ou um conceito de interesse. Uma entidade tem um conjunto de atributos que podem ter relações com outras entidades. Uma declaração de entidade define um tipo de entidade que pode ser usada para definir o domínio de um atributo.

Um **atributo** define uma importante parte de uma entidade e pode ser um atributo simples ou um atributo agregado. Um atributo é declarado por um nome e um tipo. O nome é local para a entidade, mas o tipo do atributo pode ser simples (real, inteiro, string, etc.), composto, ou de entidades.

Relacionamentos podem ser de dois tipos “é um” ou “tem um”. A relação “é um” é usada para fazer uma generalização e uma hierarquização especial entre tipos de entidades que são relacionadas no sentido em que representa uma coisa comum, e.g. um cachorro “é um” animal.

A relação “tem um” é usada para descrever uma associação entre tipos diferentes de entidades, em que um tipo de entidade pode ser considerado como parte de outro tipo, e.g. um carro “tem um” conjunto de quatro rodas.

A linguagem EXPRESS também define um mecanismo poderoso de restringir os valores de um atributo e as diferentes relações entre entidades. As regras de **restrições** podem ser aplicadas localmente para cada entidade ou para todo o esquema, de modo global [53].

A linguagem EXPRESS também pode ser apresentada em sua forma gráfica, denominada EXPRESS-G, o que a torna muito mais amigável. Atualmente, existem várias ferramentas de software para a criação de diagramas em EXPRESS-G, alguns componentes podem estão apresentados na Figura 2.9.

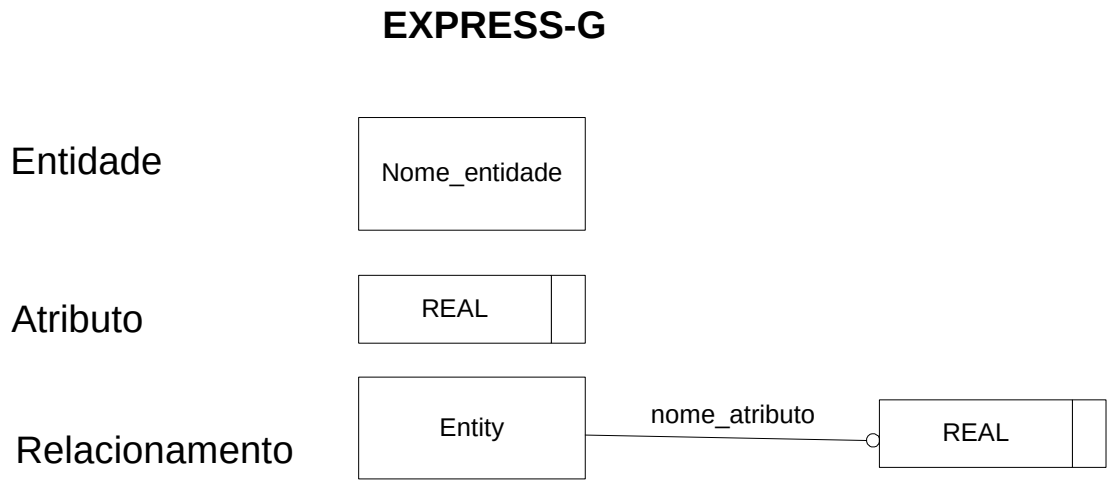


Figura 2.9: Componentes da linguagem EXPRESS-G.

Os diagramas criados em EXPRESS-G podem ser traduzidos para o modelo de informações das normas STEP ou para outros sistemas. Na Figura 2.10 é apresentada a descrição de um furo para o protocolo de aplicação AP224 em EXPRESS na forma texto, e na Figura 2.11 a mesma entidade na sua forma gráfica em EXPRESS-G.

```
ENTITY hole
 ABSTRACT SUPERTYPE OF (
ONEOF(countersunk_hole,round_hole,counterbore_hole) )
  SUBTYPE OF (machining_feature);
(*
UOF:T1
*)
  edge_radius : numeric_parameter;
 END_ENTITY;
ENTITY round_hole
 SUBTYPE OF (hole);
(*
UOF:T1*)
  diameter : circular_closed_profil;
  hole_depth : linear_path;
  change_in_diameter : OPTIONAL taper_select;
  bottom_condition : hole_bottom_condition_select;
 END_ENTITY;
```

Figura 2.10: Descrição de um fu ro em linguagem EXPRESS conforme o AP 224.

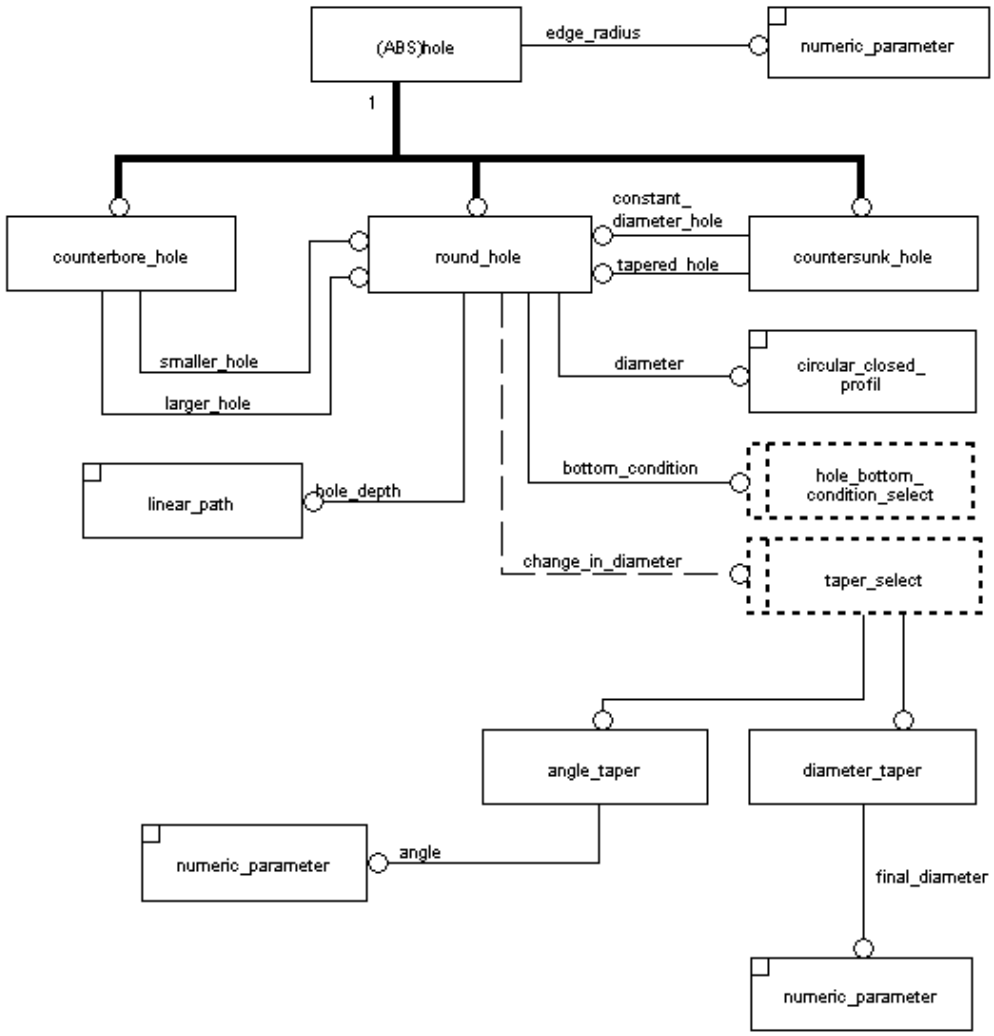


Figura 2.11: Descrição dos furos em EXPRESS-G conforme AP 224

2.6.1 Ferramentas para utilização da norma

Existem várias ferramentas para a exportação de um arquivo STEP. Em geral, estas ferramentas provêm um ambiente para a exportação de arquivos e verificação se a exportação do arquivo está de acordo com o esquema EXPRESS [54].

As tecnologias atuais de banco de dados foram totalmente utilizadas no desenvolvimento das normas STEP. As partes da norma estão organizadas em três camadas de arquitetura de dados. Esta escolha faz sentido, pois as normas STEP contêm uma enorme quantidade de informação, similar a um banco de dados.

A implementação dos bancos de dados da STEP, no entanto, é substancialmente diferente do desenvolvimento da norma STEP. A implementação envolve somente os protocolos de aplicação e os métodos de implementação. O modelo de aplicação interpretado (AIM - Application Interpreted Model), a última etapa dos protocolos de aplicações é escrito em EXPRESS. Entretanto, de modo diferente de um esquema conceitual de um sistema gestor de banco de dados, o esquema EXPRESS não serve para ser diretamente traduzido para dicionários de dados [51]. Ao invés disto, os métodos de implementação definem formatos particulares e estratégias para o uso do esquema e podem incluir exportação de arquivos, interface de programação de aplicação e implementação de banco de dados. A especificação do EXPRESS somente concerne aos aspectos formais do esquema.

O método de exportação de arquivos define a transformação do esquema em EXPRESS para um arquivo em formato texto em que para cada ocorrência de uma entidade é atribuído um número unívoco.

A classe de implementação também tem condições de traduzir a linguagem EXPRESS para linguagem de computação. Por exemplo, a tradução de EXPRESS para C++ está especificada na parte ISO 10303-23. Esta parte dá toda a funcionalidade para qualquer aplicação em STEP e especifica a construção de classes em C++ diretamente do modelo EXPRESS [47].

A interface padronizada de acesso a dados (SDAI - Standard Data Access Interface), que está descrita na parte 22 das normas STEP, é uma interface que tem como objetivo isolar os dados do produto da tecnologia específica de armazenamento de dados. Também especifica uma meta-representação da linguagem EXPRESS; esta meta-representação representa em EXPRESS a linguagem EXPRESS. A entidade chamada “schema_representation” representa um esquema; a entidade nomeada de “entity_representation” representa uma entidade; a entidade chamada “defined_type” representa um tipo e a entidade “global_rule” representa uma regra.

A SDAI também contém operadores de baixo nível para criar, apagar e reparar objetos, bem como inserir, modificar e apagar atributos. Programas de ligação em linguagens de programação específicas (C, Fortran, C++, Ada e Pascal) foram desenvolvidos de modo a permitir que as funções da SDAI acessem o esquema e

associem os dados do modelo do produto de modo independente do sistema de armazenamento de dados.

2.7 O desenvolvimento de software orientado a objetos

Para o desenvolvimento de software podem ser utilizadas várias metodologias, sendo que para o desenvolvimento de software orientado a objetos tem-se vantagem em uma abordagem dentro do paradigma da orientação a objeto, desde o início do desenvolvimento.

2.7.1 Paradigma da orientação a objetos

O paradigma da orientação a objetos é um padrão para a modelagem de sistemas. Sua utilização está historicamente vinculada à modelagem de sistemas para simulações, através da construção de programas de computador baseados nestes modelos. A primeira linguagem de programação criada com a capacidade de aceitar os modelos orientados a objetos foi a linguagem SIMULA, surgida em 1967 (Figura 2.12).

Três diferentes linguagens utilizaram os conceitos definidos na linguagem SIMULA: SMALLTALK, EIFFEL e C++, sendo que a linguagem C++ foi um aprimoramento da linguagem C e a linguagem EIFFEL é derivada da linguagem ADA.

A linguagem orientada a objetos mais atual é a linguagem JAVA derivada das linguagens C++ e SMALLTALK.

Uma das dificuldades para um programa de computador baseado em um modelo orientado a objetos é a demanda um grande espaço de memória, devido aos vários acessos e também à necessidade de um processamento muito rápido, em comparação com os programas procedurais.

Estas condições de memória e velocidade de processamento não estavam disponíveis à época da criação da linguagem da SIMULA, por isso a utilização dos modelos orientados a objetos em programas de computador só veio a tomar impulso na década de 1990, especialmente após a criação da linguagem de

programação C++, que foi um aperfeiçoamento da linguagem C, englobando as possibilidades de se trabalhar com modelos orientados a objetos [55].

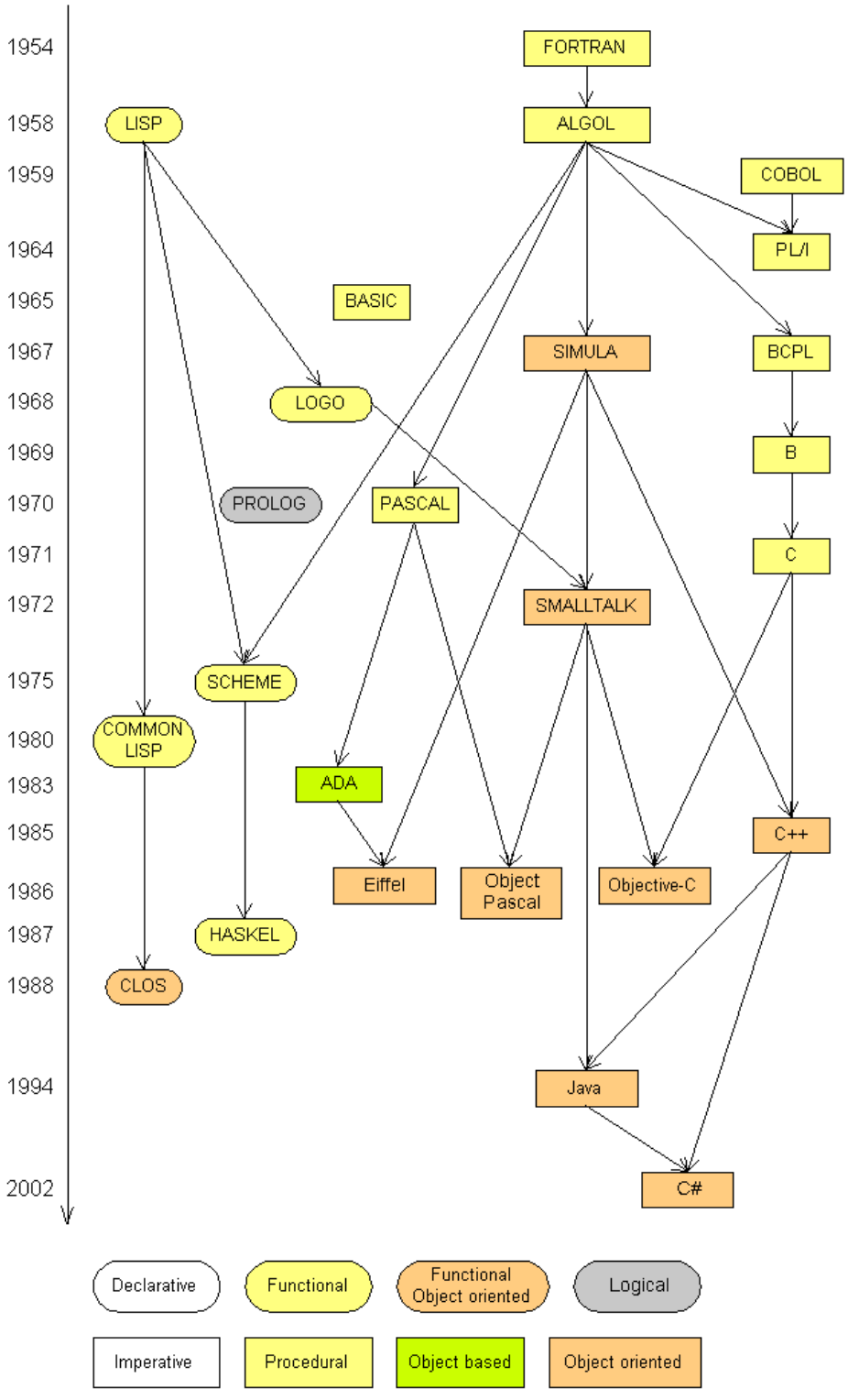


Figura 2.12: Quadro histórico do desenvolvimento de linguagens de programação [56].

Em uma abordagem procedural, a criação de um modelo de um sistema real é feita através da decomposição do sistema em vários processos, que são as ações realizadas pelos componentes do sistema na transformação de entradas em saídas.

A criação do modelo é feita através da definição dos processos internos ao sistema. Em uma abordagem orientada a objetos, o foco da decomposição está nos componentes utilizados para converter entradas em saídas. O modelo é criado através das definições dos componentes que realizam processos.

No paradigma da orientação a objetos, o sistema é modelado como uma coleção de unidades autônomas que agregam, em si mesmas, informações (dados) e ações (métodos).

A criação de modelos dentro do paradigma da orientação a objetos é feita por meio, não de uma evolução ou aprimoramento de linguagens de programação, mas, antes de tudo, através de uma nova forma de pensar, analisar e modelar um sistema real.

Algumas das linguagens de programação desenvolvidas com a capacidade de trabalhar com modelos orientados a objetos são: C++; JAVA; ADA; Smalltalk; SIMULA.

As principais características de um programa de computador baseado em um modelo orientado a objetos são, a reutilização, a completitude, a robustez.

A reutilização é a característica do software de poder serem usadas porções de código em novos programas, ou seja, um mesmo código pode ser utilizado várias vezes.

Um modelo orientado a objetos tem esta característica, derivada da característica de encapsulamento, pois cada objeto é completo em si mesmo e independente de qualquer outro. Desta forma, um objeto pode ser inserido em qualquer código.

Devido ao desenvolvimento inicial de um programa baseado em um modelo orientado a objetos ser a construção do modelo, o programa tende a ser completo, ou a pelo menos permitir a construção parcial com a visão do todo.

A robustez de um software é a característica da ausência de falhas na sua utilização. Devido à impossibilidade de elaboração de um teste completo a

verificação da robustez de um software é feita durante seu uso. Por utilizar um modelo completo e por ser cada objeto pensado e resolvido por si, um software orientado a objeto tende a ser bastante robusto.

A orientação a objetos permite o gerenciamento de informações (dados) e funções dentro de um mesmo modelo. Num modelo orientado a objetos, somente existem objetos, nenhuma outra estrutura é utilizada, e nesta estrutura estão combinados dados e funções. Os objetos são encapsulados de tal maneira a terem “auto-responsabilidade” sobre as informações armazenadas e sua consistência.

A utilização do paradigma da orientação a objetos permite a criação de um modelo que englobe em si uma estrutura de dados com a capacidade de conter informações tecnológicas e geométricas

No caso de um modelo baseado em *features*, a grande facilidade está na vasta similaridade de conceitos, cada *feature* é um objeto.

A construção de um modelo baseado em *features,* através do paradigma da orientação a objetos, cujo objetivo final é a integração do desenvolvimento do produto, parte da determinação das classes, atributos e métodos que irão prover o encapsulamento dos objetos, para maximizar a reutilização de código e armazenar todas as características necessárias para a manipulação dos modelos em todas as fases do desenvolvimento do produto. Estes requisitos podem ser assim classificados:

- O modelo deve permitir a integração de todas as ferramentas de desenvolvimento do produto;
- O modelo deve ser independente da plataforma e da linguagem;
- O código deve permitir a reutilização.

O projeto feito dentro do paradigma da orientação a objetos permite que o sistema seja ampliado com facilidade, incorporando classes necessárias a esta ampliação e membros relevantes, sem alteração de sua estrutura geral [57].

2.7.2 A linguagem de modelagem unificada - UML

A formulação de um modelo implica na utilização de uma linguagem capaz de descrever este modelo de maneira adequada. Desde a criação do paradigma da orientação a objetos, várias notações específicas foram utilizadas, dentre as quais se destacam: BOOCH; OOSE; OMT-2; FUSION.

Em 1994, dois autores, Booch e Rumbaugh, resolveram aglutinar suas notações e criar um método unificado. Um ano depois, estando o método já na versão 0.8, Ivar Jacobson integra o grupo e após a revisão dos objetivos surge a linguagem de modelagem unificada - UML [4].

No ano seguinte, foi lançada a versão 0.9, contando com várias contribuições externas. Em 1997, a versão 1.0 da UML foi apresentada a OMG, que, em 14 de novembro de 1997, adotou, já a versão 1.1, como padrão para modelagem de sistemas orientados a objetos.

A UML é uma linguagem que pode ser utilizada para especificar, visualizar, construir e documentar sistemas através de modelos orientados a objetos, tendo como principais características:

- Não proprietária, com código-aberto;
- Tem mecanismos extensíveis;
- É independente de linguagens de programação ou de processos de desenvolvimento de software.

A UML utiliza diagramas para a construção de modelos e estes diagramas são utilizados nas diversas fases do processo de desenvolvimento do software.

Para o processo de desenvolvimento de software, a UML disponibiliza 37 diferentes diagramas. Durante o desenvolvimento do projeto *FESTEVAL,* foram utilizados os seguintes:

- Diagramas de Casos de Uso;

  São diagramas usados para identificar o comportamento do sistema em várias situações que podem ocorrer durante a operação, descrevem o

sistema, o ambiente e o relacionamento entre os dois. São compostos de atores e casos.

- Diagramas de Seqüência;

  Mostram a interação entre os objetos ao longo do tempo. Apresentam os objetos que participam da interação e a seqüência de informações trocadas.

- Diagramas de Classes;

  Apresentam as classes com seus métodos e atributos e os relacionamentos entre as classes, como heranças, agregações, comunicações. Este diagrama é estático e o comportamento dinâmico do modelo não faz parte deste diagrama.

- Diagramas de Pacotes;

  O diagrama de pacotes tem o objetivo de facilitar a visão geral do modelo através do agrupamento em pacotes de classes pertencentes a um subgrupo.

2.7.3 UML para as normas STEP

Nas normas STEP, o modelo de dados do produto é criado dentro do paradigma da orientação a objetos, com a utilização da linguagem de modelagem EXPRESS e sua notação gráfica EXPRESS-G.

No entanto, o uso dos diagramas da UML para as normas STEP está em discussão no ISO/TC184/SC4. Uma de suas maiores vantagens é a possibilidade de transcrição do modelo em qualquer linguagem de programação orientada a objetos [58]. Os diagramas da UML também têm a facilidade de serem lidos e escritos, por empregarem uma notação gráfica. A escolha da linguagem de modelagem a ser utilizada depende do projeto. A linguagem EXPRESS é bastante útil na transcrição dos modelos ARM e AIM e no planejamento entre diferentes protocolos de aplicação das normas STEP, entretanto o uso da UML, possibilita a transcrição direta para uma linguagem de programação orientada a

objetos [51]. Este foi o procedimento adotado no projeto *FESTEVAL*, em que o modelo em EXPRESS foi escrito em UML em diagramas de classe, e depois, com o uso da ferramenta CASE Rational Rose, transcrito para C++.

2.7.4 Banco de Dados Orientado a Objetos

Apesar de ser uma tecnologia já estabelecida com várias aplicações em uso e ter uma linguagem específica (SQL *Structured Query Language*) que atende muito bem os processos de consulta, atualização e gerenciamento de bancos de dados relacionais, estes nem sempre são úteis para aplicações de engenharia. Os principais problemas do uso de banco de dados relacionais em aplicações de engenharia são [59]:

- Perda expressiva da capacidade de modelagem;
- Incompatibilidades nas transferências de informações;
- Perda de capacidade para grandes transferências;
- Problemas de desempenho com o desenvolvimento dos sistemas e do aparecimento de novas versões.

Na sua essência, os sistemas para gerenciamento de bancos de dados relacionais utilizam um controle muito específico sobre os dados por causa dos vínculos existentes no diretório de dados, o que implica em uma implementação inicial bastante trabalhosa, mas que pode ser compensada pelo fato de ser possível trabalhar desta maneira com muitos dados; desde que as operações feitas com estes dados também não sejam numerosas. Ocorre que isto não é o caso para as aplicações de engenharia, nas quais o volume de operações com as informações é normalmente muito superior ao volume de dados armazenados, justamente o oposto das aplicações comerciais para as quais é mais indicado o uso de banco de dados relacionais [2],[54].

Os bancos de dados orientados a objetos, por outro lado, tem uma tecnologia mais adequada para atingir as necessidades das aplicações de engenharia, com as seguintes características, que os fazem eficazes, quando o volume de

operações com as informações é muito grande e quando há um contínuo desenvolvimento do modelo [60]:

- Modelagem rica de dados;
- Compartilhamento de objetos;
- Permite a evolução em objetos e classes;
- Apropriados para transações em trabalho de projeto corporativo;
- Armazenamento de objetos distribuídos e independentes da plataforma.

3 Objetivos e metodologia

A proposta de trabalho deste projeto consiste em apresentar melhorias em um modelador baseado na tecnologia *features* e desenvolver, para este modelador, uma interface STEP que, através de um banco de dados orientado a objetos, armazene as informações do modelo de acordo com as normas STEP, de forma a garantir a integração entre todas as etapas do ciclo de desenvolvimento do produto. O desenvolvimento teórico e a implementação desta proposta, conforme apresentados nos capítulos 4 e 5 visa a atingir os objetivos abaixo.

3.1 Objetivos Gerais

Considerando-se os aspectos mencionados anteriormente, os objetivos gerais deste projeto são:

- Desenvolver e implementar contribuições para um modelador para sistemas CAD baseado na tecnologia de *features*;
- Implementar o armazenamento do modelo criado de acordo com as normas STEP, visando a integração digital das informações durante o ciclo de desenvolvimento do produto.

3.2 Objetivos Específicos

Para atingir os objetivos gerais, alguns objetivos específicos têm que ser alcançados:

- Desenvolvimento de software orientado a objetos dentro do ambiente do sistema CAD Unigraphics em linguagem de programação C++, para acrescentar a este sistema as características de modelagem baseada na tecnologia de *manufacturing features*, incluindo a modelagem semântica e tecnológica e a representação das interdependências.
- Desenvolvimento do banco de dados orientado a objetos para o armazenamento das informações do modelo, utilizando para isto o

conjunto de normas STEP. Como o objetivo deste sistema de armazenamento de informações é a integração entre as etapas de desenvolvimento do produto, foi escolhido um aprimoramento do protocolo de aplicação AP224, com a capacidade de representação das interdependências.

3.3 Metodologia

O desenvolvimento do software que pretende atingir os objetivos foi dividido em quatro etapas: análise, projeto, desenvolvimento e teste, com a utilização das seguintes ferramentas de software para as implementações descritas acima:

- **UG/Open** - Biblioteca de classes em C++ para acesso às funções do software Unigraphics, possibilitando a criação de APIs.

- **NIST STEP Class Library** - Biblioteca de classes em C++ para reutilização de software, possibilitando a criação automática de uma biblioteca de classes para um protocolo de aplicação das normas STEP e a transferência dos dados obtidos para um arquivo físico;

- **WinSTEP** - Software para utilização da biblioteca de classes do NIST;

- **Infomodel** - Biblioteca de classes que controla o uso da biblioteca de classes do NIST;

- **AP224_IM** - Biblioteca de classes que dá condições de registrar e recuperar o arquivo físico com os dados do produto, de acordo com as normas STEP. Neste projeto, esta biblioteca de classes será a única interface de acesso ao banco de dados do produto;

- **Rational Rose Tools** - Conjunto de ferramentas que possibilitam o uso dos diagramas UML e a manipulação destes na forma de código-fonte.

4 Desenvolvimento Teórico

O sistema CAD é o elo inicial na cadeia de desenvolvimento projeto-processo-manufatura. A qualidade e o formato da informação fornecida pelo CAD pode contribuir para construir uma cadeia de desenvolvimento de processo realmente integrada digitalmente ou para construir uma barreira de informações e interpretações. A questão central é a semântica e a estrutura utilizada para a representação da informação de projeto, isto é, é necessário adotar uma linguagem que possa ser entendida por todos os sistemas CAx subseqüentes da cadeia de desenvolvimento (CAPP, CAM, CAQ, etc.).

4.1 A Criação do modelo por *FEATURES*

A modelagem baseada em *form features* é um suporte ativo para os projetistas, pois sua aplicação permite o destaque do conhecimento envolvido no processo. Através da introdução de conhecimento de manufatura nas *form features* (*manufacturing features*), por exemplo, é possível verificar a manufaturabilidade da peça ainda durante na fase de projeto, contribuindo efetivamente para a introdução de um processo de engenharia simultânea. Seguindo este conceito, o projeto será baseado em uma biblioteca de classes de *manufacturing features*.

São encontradas várias definições do conceito de *form features* na literatura, inclusive algumas delas com conflitos de definições. Porém, é importante definir o entendimento de *form feature* neste trabalho. A definição para este trabalho é:

> ***Form feature*** *é um objeto que engloba uma semântica, uma parametrização, uma representação geométrica e informações tecnológicas (interdependências e atributos). Figura 4.1*

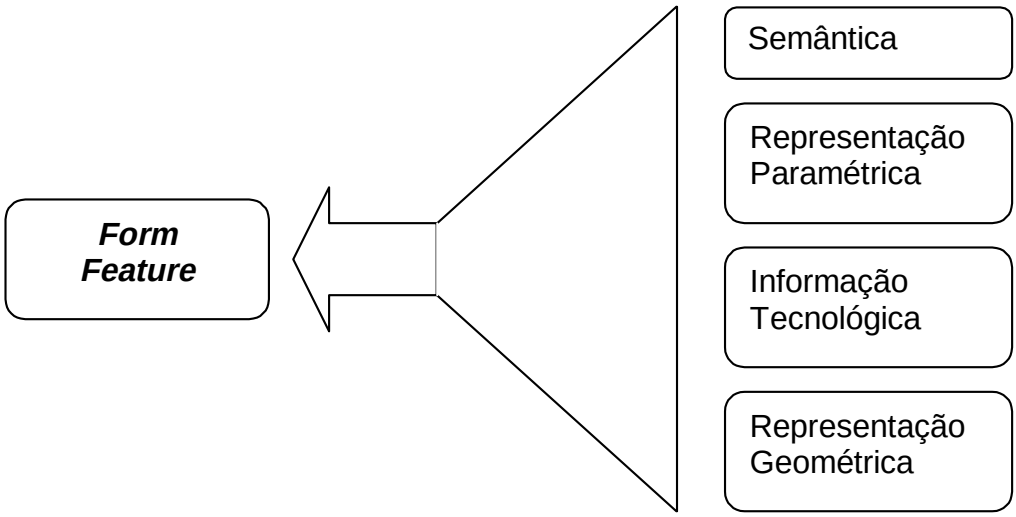


Figura 4.1: Definição de form feature [25].

Neste trabalho, o objetivo é a integração do sistema CAD com os sistemas CAx subseqüentes e trazer para a fase de projeto as restrições de manufatura que irão caracterizar as *features* de projeto em *manufacturing features.* Nesta aplicação específica é que se caracteriza a semântica das *form features.*

A informação de conhecimento tecnológico faz a *feature* "inteligente". Esta informação possibilita que, no momento em que uma *feature* é inserida no modelo durante o projeto, métodos de validação são acionados sendo possível, entre outras, a verificação da manufaturabilidade. Através das *features* é provido o auxílio aos projetistas na introdução de tolerâncias, na especificação de tratamento térmico, etc.

A biblioteca de classes, utilizada para a construção do modelo, contém *features* parametrizadas, cujos valores são inseridos para que a *feature* faça parte do modelo.

Enquanto classes, as *manufacturing features,* existentes na biblioteca, contêm:

- Uma semântica orientada à manufatura;
- Um único conjunto de parâmetros;
- Um método de construção baseado em seus parâmetros para gerar a representação geométrica;

- Os elementos da *feature* (*faces* e *edges*);
- Conhecimento de manufatura;
- Características tecnológicas (tolerâncias, acabamento superficial, etc.).

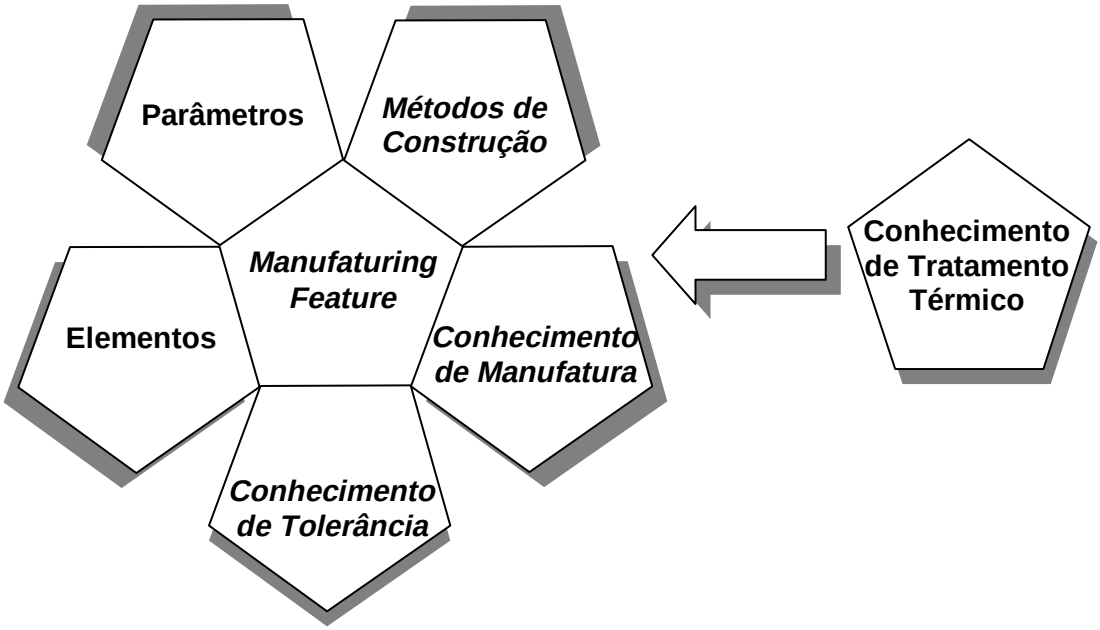


Figura 4.2: Características de uma manufacturing feature [25]

Toda informação tecnológica de uma *manufacturing feature* que permite, por exemplo, dar apoio ao projetista para a definição de tolerâncias, como também os métodos implementados para validar uma *manufacturing feature*, para verificar sua manufaturabilidade, etc., está relacionada à semântica da *manufacturing feature.*

Os módulos subseqüentes do projeto *FESTEVAL* também possuem métodos que estão relacionados à semântica das *manufacturing features*.

Para a determinação da ferramenta apropriada, da estratégia de corte, dos parâmetros de corte, etc., existem os métodos apropriados para cada classe de *features*, ou seja, de acordo com a semântica da *feature*. Por exemplo, quando os módulos de planejamento e manufatura devem realizar a operação de usinagem

de um *rectangular pocket*, são escolhidas uma fresa de topo como ferramenta e como estratégia de usinagem um percurso em espiral, considerando a informação encapsulada na semântica desta *feature*.

4.2 O Armazenamento das Informações

As informações do modelo precisam estar armazenadas de modo a serem recuperadas pelo protótipo e disponibilizadas para a criação do arquivo físico STEP, o modelo de dados usado neste projeto é uma adaptação advinda da composição três protocolos de aplicação e de entidades criadas especialmente para o projeto *FESTEVAL* [61], a composição deste modelo de dados é apresentada na Figura 4.3 Este modelo de dados define atributos que não fazem parte do modelo de dados do UG, como por exemplo, as interdependências entre *features*. Também alguns atributos, apesar de estarem disponíveis no modelo de dados do UG não estão de forma adequada ao modelo de dado provido pelas normas STEP, como tolerâncias dimensionais.

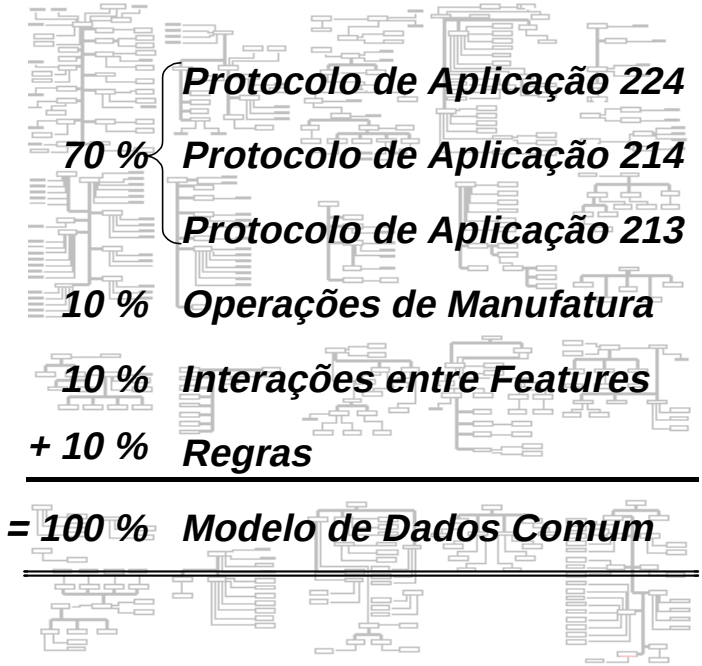


Figura 4.3: Composição do modelo de dados [61].

Depois de ser criada, no ambiente *FESTEVAL*, somente os atributos que estão definidos no UG podem ser armazenados no banco de dados do UG.

As informações das *features* podem, então, ser divididas em dois grupos, informações geométricas que fazem parte do modelo de dados do UG e todas as outras informações que não fazem parte do modelo de dados do UG. As primeiras podem ser facilmente armazenadas no banco de dados do UG, as últimas precisam ser armazenadas de outra forma. Uma facilidade do modelador utilizado é a possibilidade de estender o banco de dados através da criação de atributos. Desta forma não há necessidade de utilização de banco de dados redundantes nem de utilização de dois bancos de dados.

4.3 Validação Semântica e Reconhecimento das Interdependências

Durante o desenvolvimento do planejamento de processo uma das principais atividades é determinar a seqüência ótima de operações de manufatura. Para esta tarefa é fundamental, não apenas obter a representação geométrica das *manufacturing features* da peça, mas também suas semânticas e as interdependências entre elas. Apenas considerando estas características é possível determinar a correta seqüência de manufatura.

Um exemplo típico de uma interdependência que influencia na manufatura é um furo posicionado na base de um *pocket*. Existem seqüências de manufatura diferentes para usinar as duas *features*, porém o sistema de planejamento de processo necessita das informações sobre as interdependências entre as duas *features* para tomar a decisão correta.

Um exemplo da necessidade de se verificar a validade semântica das *features* é a edição dos parâmetros de um *rectangular pocket* que, se forem modificados durante o projeto, pode existir a necessidade de se modificar também a forma de usinagem da peça. Um *rectangular pocket* fechado, por exemplo, só pode ser usinado com uma fresa de topo. Se seus parâmetros forem modificados, resultando em uma *manufacturing feature* s*lot*, ou se seus parâmetros e posicionamento forem modificados, resultando em uma *manufacturing feature step,* veja Figura 4.4, a usinagem poderá ser feita com fresa de topo ou com uma fresa de disco, pois existem diferentes acessos para a ferramenta. Sem validar a semântica da *feature* e modificá-la de acordo com as alterações de parâmetros, os módulos de planejamento e produção escolhem, como antes, o mesmo tipo de ferramenta e estratégia de corte, que não são mais ideais para a *manufacturing*

feature real (*slot* ou *step*). Existem meios mais produtivos para usinar estes outros tipos de *manufacturing features.*

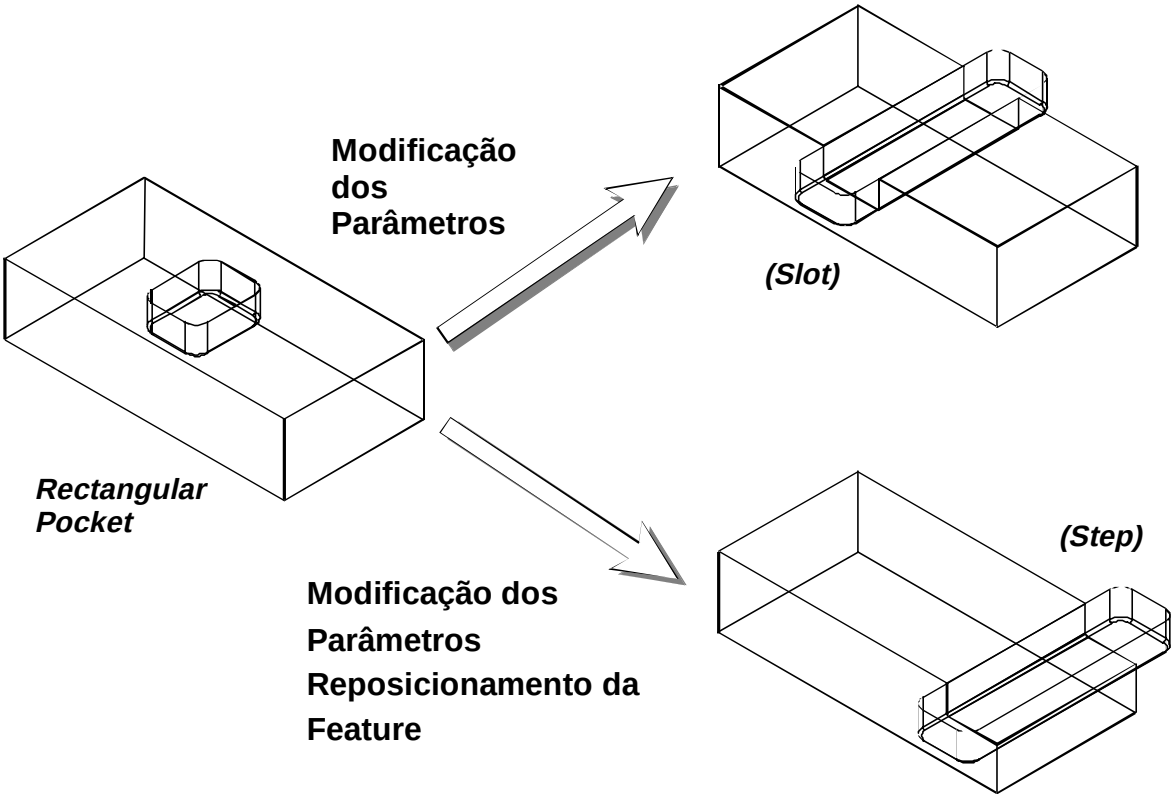


Figura 4.4: Modificações em uma manufacturing feature Retangular Pocket

Portanto, é necessário garantir que a semântica e a representação da *feature* permaneçam coerentes durante a cadeia de desenvolvimento do produto (projeto-processo-manufatura) para que a fabricação da peça seja otimizada.

Considerando os problemas citados acima, tem-se o conceito de persistência da semântica da *manufacturing feature*. De acordo com isso, devem ser implementados métodos para cada classe de *manufacturing features*, que represente regras de validação para verificar a manutenção da semântica da *feature* ao longo do projeto.

Por exemplo, um *open pocket* possui as seguintes regras de persistência semântica:

- Regra 1: deve possuir cinco faces cilíndricas;

- Regra 2: todas as faces materiais devem estar a uma distância mínima do objeto inicial;
- Regra 3: a face virtual superior é paralela à face inferior;
- Regra 4: a face virtual lateral é perpendicular à face inferior.

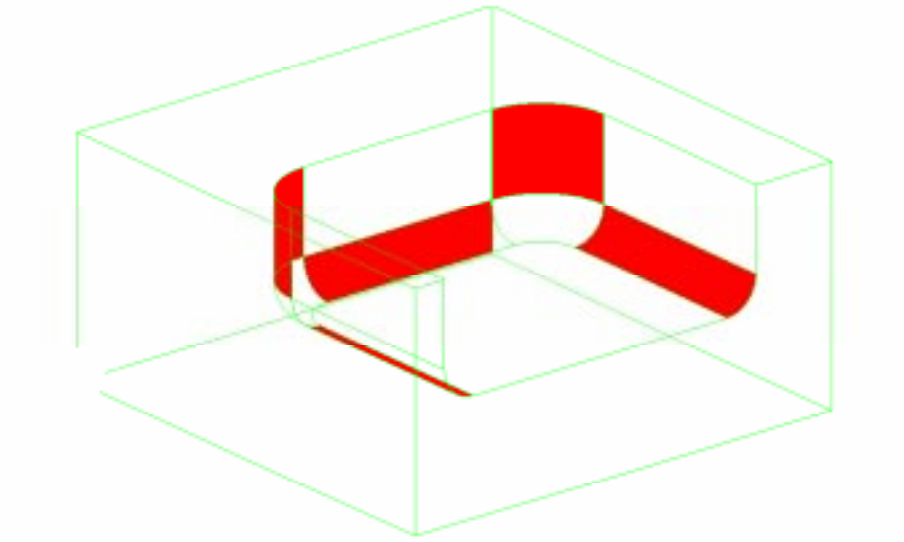


Figura 4.5: Regra 1- Possuir cinco faces cilíndricas

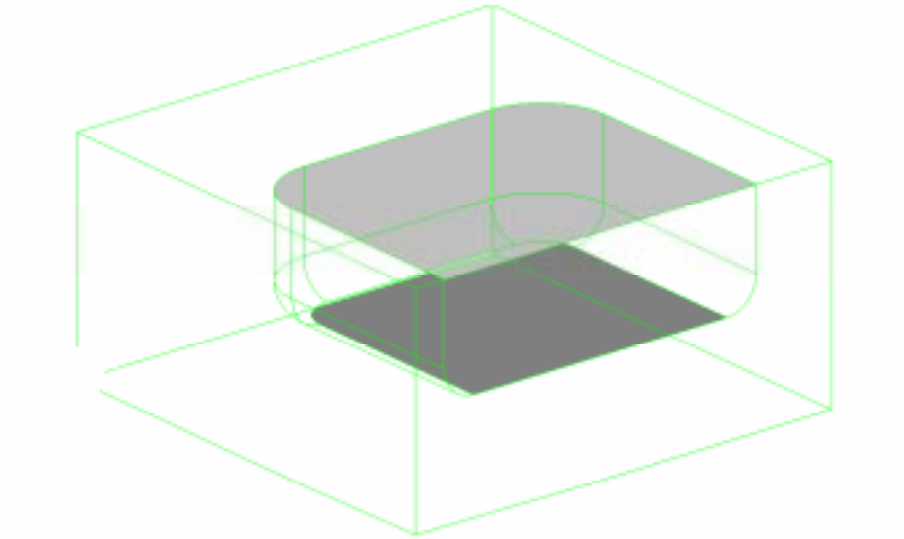


Figura 4.6: Regra 3 - A face virtual superior é paralela à face inferior

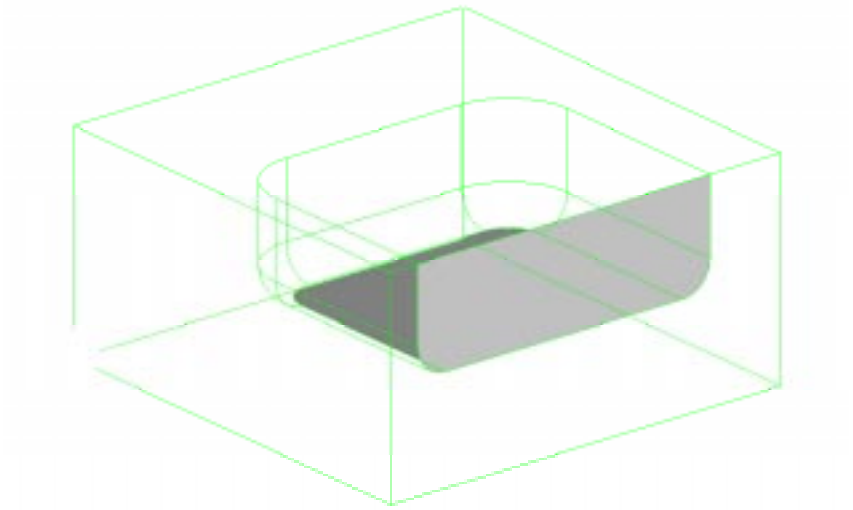


Figura 4.7: Regra 4 - A face virtual lateral é perpendicular à face inferior.

4.4 Classificação das Interações

O reconhecimento, validação e representação das interações entre *manufacturing features* no modelo de informações através do ambiente de projeto baseado em *features* são os pré-requisitos para o suporte baseado em *features* ao processo de desenvolvimento da cadeia projeto-processo-manufatura [62].

Os sistemas CAPP não possuem um modelador geométrico, sendo necessário que o desenvolvimento do projeto baseado em *features* auxilie o CAPP quando alguma espécie de característica geométrica é necessária, para tomar uma decisão de usinagem.

Por exemplo, a escolha do tamanho das ferramentas de acordo com a geometria e dimensões da peça. A Figura 4.8 mostra alguns destes problemas que o planejamento de processo deve resolver; note que muitas destas situações também estão relacionadas com as interações entre *features*.

Nos casos *a* e *b,* o CAPP é confrontado com uma passagem estreita que deve ser levada em consideração para escolher o correto diâmetro da ferramenta. Nos casos *c* e *d,* existem outros dois casos de interdependências entre *features* e o CAPP necessita da completa informação sobre as *features*, incluindo as restrições de diâmetro de ferramenta, para evitar uma colisão com a parede da segunda *feature* e a usinagem incompleta da primeira *feature*.

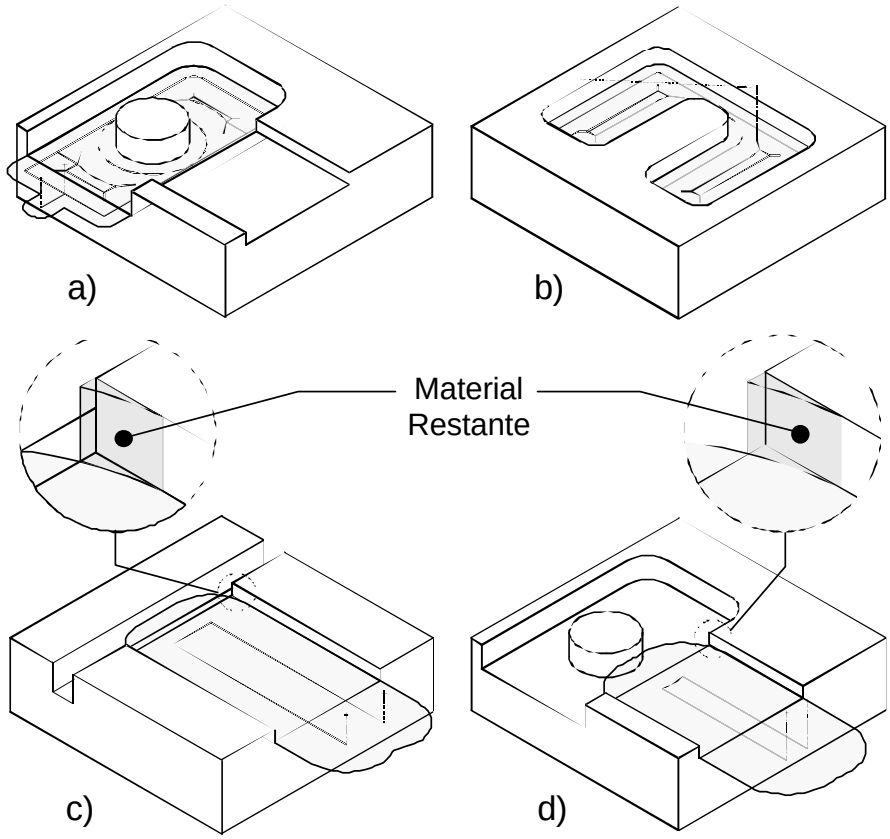


Figura 4.8: Exemplos de restrições de manufatura: a) interação de volume; b) pocket côncavo; c) interdependência precedente; d) interdependência tecnológica implícita

As interações tecnológicas resultam da definição de alguns atributos tecnológicos e as interações geométricas resultam de características geométricas das *manufacturing features* e seu posicionamento relativo na peça.

Uma análise detalhada das interações geométricas entre *manufacturing features* permite classificá-las em dois grupos principais: interações por face e interações por volume, conforme apresentado em 4.4.2.

As interações tecnológicas, por sua vez, indicam o compartilhamento de uma restrição tecnológica, como é o caso de uma tolerância de concentricidade entre dois furos para mancais em uma caixa de engrenagens.

4.4.1 Interações Tecnológicas

As interações tecnológicas estão relacionadas à especificação de atributos tecnológicos. Eles podem ser: gerais ou baseados em *features*. Tem-se por

atributos tecnológicos gerais, o material, tolerâncias gerais e tratamento térmico para a peça toda. Os atributos tecnológicos, baseados em *features,* compreendem tolerância dimensional, tolerância geométrica, tolerância de posição, qualidade superficial e tratamento térmico aplicado a um elemento de superfície de uma *manufacturing feature*.

O primeiro tipo de atributos é definido para toda a peça e não resulta em interações entre *manufacturing features* específicas. Alguns dos atributos tecnológicos gerais, como perpendicularidade e simetria, impõem adicionalmente uma interdependência, entretanto, também neste caso, não estão relacionados a uma *manufacturing feature* específica, mas a todas as superfícies perpendiculares ou objetos simétricos. Portanto, este tipo de interdependência “geral” é representado como uma característica genérica para toda a peça.

Os atributos tecnológicos baseados em *features* são adicionalmente divididos em dois grupos. No primeiro, estão os atributos tecnológicos relacionados a um parâmetro ou a um elemento de apenas uma *manufacturing feature*. No segundo grupo, estão os atributos tecnológicos que definem uma relação entre elementos de duas ou mais *manufacturing features*.

Um exemplo é a tolerância dimensional, que deve ser dividida em dois tipos: o primeiro tipo, tolerância dimensional interna, é aplicado a um parâmetro de uma *manufacturing feature*; o segundo tipo, tolerância dimensional externa, define a tolerância para uma medida entre dois elementos paralelos de duas *features* diferentes e, portanto, define uma interdependência.

Resultando das considerações descritas acima, o primeiro grupo de atributos tecnológicos inclui: tolerância dimensional interna, tolerância geométrica, qualidade superficial e tratamento térmico para um elemento de superfície de uma *manufacturing feature*. Estas informações se referem a um parâmetro ou a um elemento de uma *manufacturing feature* no Modelo de Informações.

Os atributos tecnológicos do segundo grupo são: tolerância dimensional externa e tolerância de posição. Estes atributos são aplicados em um elemento de uma *manufacturing feature* em relação a um ou mais elementos de outras *manufacturing features*, portanto, resultam em interdependências tecnológicas , automaticamente reconhecidas e gerenciadas pelo Módulo de Projeto.

4.4.2 Interações Geométricas

Na literatura é possível encontrar referências a interdependências geométricas entre *features*. Contudo, o foco principal destes trabalhos é a geometria e a topologia da peça. São considerados problemas, a divisão de faces e a construção de novos *edges*, como no caso da interseção de dois *slots* perpendiculares ou a interseção de um furo com um *pocket*.

O foco deste trabalho é o reconhecimento, a representação e o gerenciamento de interações do ponto de vista da usinagem. A decisão sobre o posicionamento das *manufacturing features* durante o projeto pertence aos projetistas; o protótipo deve validar cada *manufacturing feature* considerando seus parâmetros, elementos e manufaturabilidade.

A interação geométrica é originada durante o processo de projeto através da criação das *manufacturing features* disponíveis na biblioteca de *features* e se refere, no mínimo, a duas *manufacturing features*. Neste trabalho, elas são classificadas em dois grupos, de acordo com o tipo de *feature* envolvida na interação.

4.4.2.1 Interação por Volume

As interações por volume ocorrem quando um volume pertence simultaneamente a duas *features*. Na execução da peça este volume comum pode ser usinado por qualquer uma das *features*. Na Figura 4.9, o rebaixo e o furo têm um volume comum, este volume pode ser usinado no momento de execução do furo ou no momento de execução do rebaixo, neste caso o modelo deve preservar a informação da ordem de inserção das *features* no modelo.

O compartilhamento de um volume pode acarretar na invalidação semântica da *feature*, por isso a verificação das interações deve estar acompanhada da validação semântica.

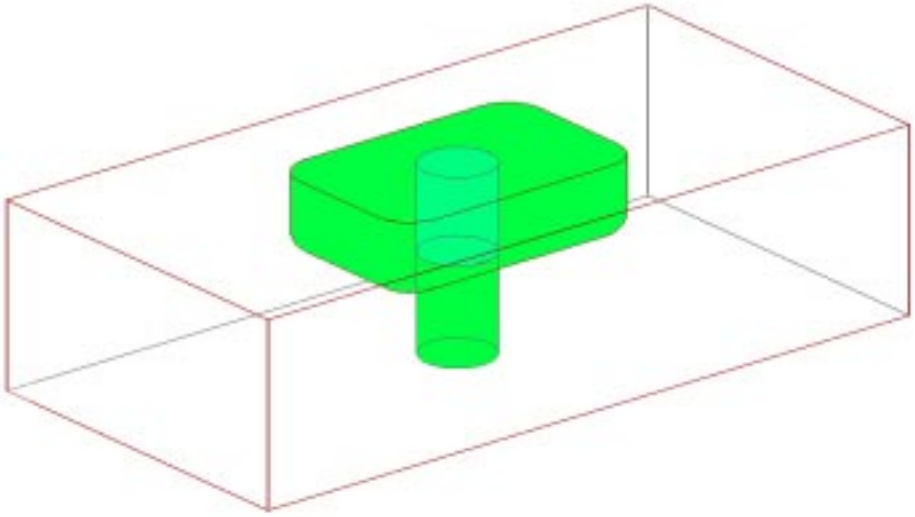


Figura 4.9: Interação por volume

4.4.2.2 Interações por Face

As interações por face são aquelas que ocorrem quando duas *features* compartilham da mesma superfície, causando uma relação de interdependência entre as *features*. A face compartilhada só existe após a execução de uma *feature* que, portanto, tem precedência sobre a outra *feature*, ver Figura 4.10. O reconhecimento da interação entre as *features* é feito através da análise das arestas de cada *feature*.

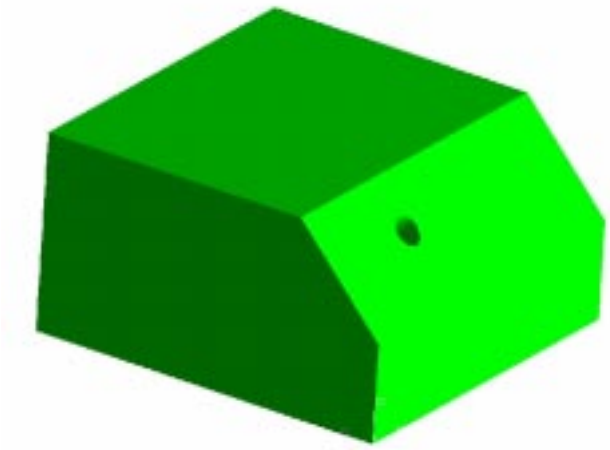


Figura 4.10: Interação por face com interdependência.

Como a interação superficial está relacionada à ordem de execução das *features,* sua determinação ocorre pela ordem de inserção das *features* no modelo.

4.4.2.3 Interação geométrica implícita

Interdependências geométricas implícitas são as resultantes de uma restrição implícita entre *manufacturing features*.

Este tipo de interdependência pode, por exemplo, ser definido através de uma característica geométrica. Este é o caso da parede mínima entre duas *features*, como também através de uma característica tecnológica, que é o caso da interação produzida durante a geração do caminho da ferramenta NC para usinar uma *feature*.

Atributos definidos para *features* são derivados de condições geométricas de manufatura ou da utilidade de cada *feature.* As regras de manufatura resultam em atributos de natureza física, geométrica ou topológica que podem estar de acordo com um estudo conceitual ou considerações práticas de usinagem.

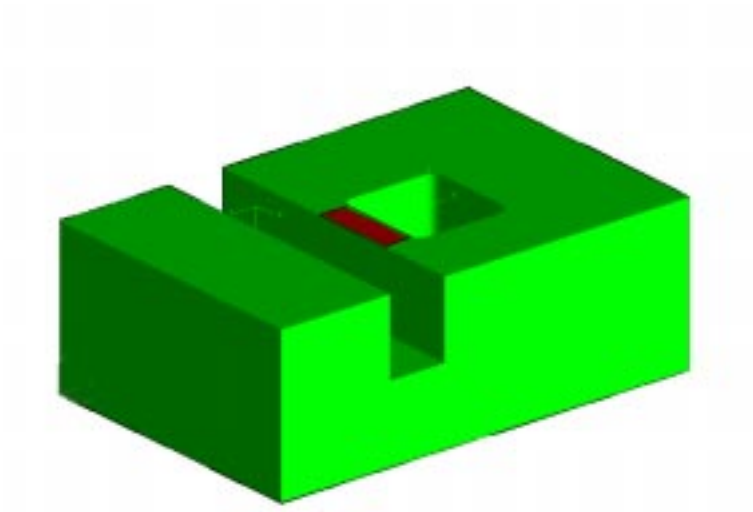


Figura 4.11: Interação implícita por espessura mínima de parede.

Um atributo de uma *feature* pode causar um vínculo de interdependência com outra *feature*, ou com um atributo de outro *feature*. Os exemplos abaixo ilustram interdependências implícitas no projeto e na manufatura. No caso da Figura 4.11,

é apresentada uma interdependência entre duas *features* devido à espessura de parede entre elas. No caso da Figura 4.12, a interdependência se dá pelo caminho a ser percorrido pela ferramenta na execução das *features*. As interdependências implícitas são de reconhecimento muito difícil e podem afetar o desenvolvimento do produto em qualquer de suas etapas.

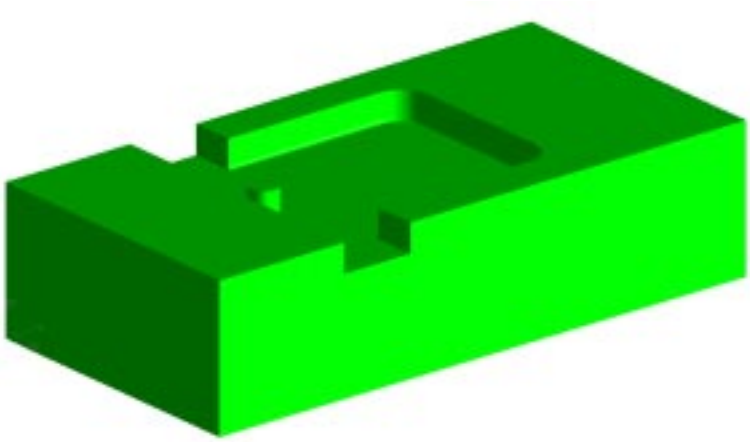


Figura 4.12: Interação implícita por caminho de ferramenta.

Este tipo de interação também pode ter influência de outros atributos não geométricos. Este é o caso da parede mínima mencionada, cujo valor crítico depende do material e das condições de usinagem. Como também, a interação do caminho da ferramenta NC que pode ser eliminado através da escolha de uma ferramenta de menor diâmetro.

4.5 Atributos Tecnológicos

O ponto inicial para a especificação dos atributos tecnológicos, que serão considerados ao longo do projeto, será a necessidade do usuário final, representado pelas Indústrias Romi S.A. [63]. Contudo, a intenção é olhar para solução genéricas, o que irá atender outros projetistas na modelagem de peças prismáticas.

A generalização das necessidades do usuário final para atributos tecnológicos de peças prismáticas baseadas em *manufacturing features* resulta em dois principais grupos de atributos:

- Atributos tecnológicos gerais, os quais estão relacionados com toda a peça, como especificação de material, tolerâncias gerais, tratamento térmico para a peça, etc.

- Atributos tecnológicos baseados em *manufacturing features*, os quais estão relacionados a um parâmetro da *feature* (comprimento, diâmetro, profundidade, etc.) ou a um elemento da *feature* (face lateral de um *slot*, face cilíndrica de um *hole*, etc.). Eles podem ser tolerância dimensional, tolerância geométrica, tolerância de posição, qualidade superficial, tratamento térmico para um elemento da *feature*, etc.

O Modelo de Informações do *FESTEVAL* considera todos estes tipos de atributos e os representa, como apresentado no item 5.4.3. A Figura 4.13 mostra uma das peças de referência fornecidas pelas Indústrias Romi S.A. [63] com os atributos mencionados. Dois dos atributos tecnológicos, a cilindricidade e a perpendicularidade, não existem na peça real, pois foram incluídos apenas para representar estes atributos que foram implementados no protótipo.

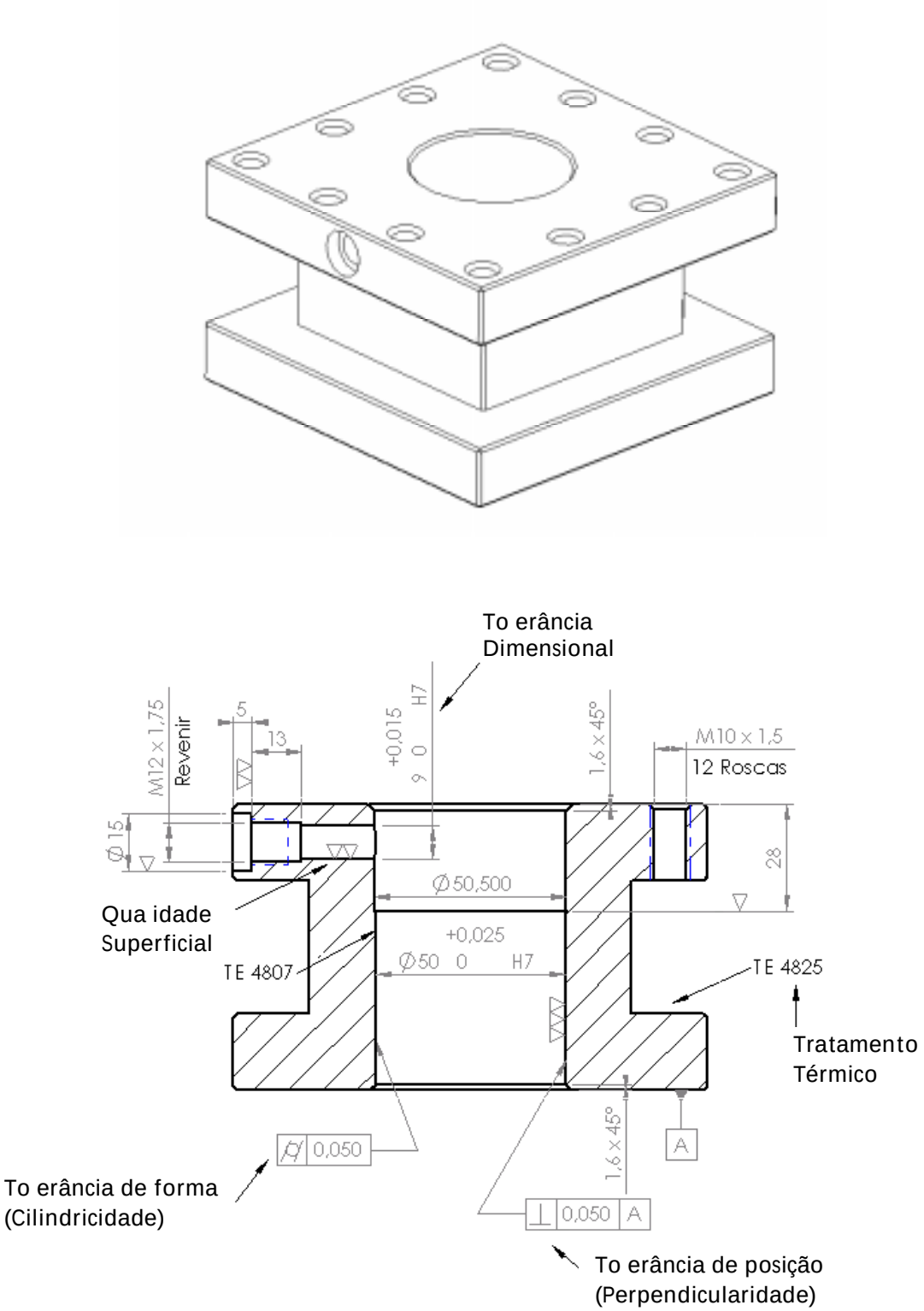


Figura 4.13: Peça referência com alguns atributos tecnológicos [63].

4.5.1 Atributos Tecnológicos Gerais

Conforme mencionado no tópico 4.5, a especificação do material é um tipo de atributo tecnológico geral que deve ser definido pelo projetista da peça. O protótipo deve fornecer um banco de dados de materiais com todas suas especificações, dos quais os projetistas podem escolher o mais apropriado para a peça durante o processo de projeto. Este banco de dados deve ser preenchido com os materiais específicos utilizados na empresa, o que permitirá aos projetistas ter uma rápida visão dos materiais disponíveis nas normas internas da empresa.

Para os testes e validações do protótipo, o banco de dados de materiais será preenchido com alguns materiais de acordo com as normas ISO, embora devam estar disponíveis funções do banco de dados, o que permitirá a cada empresa configurar o banco de dados de acordo com suas normas internas.

O banco de dados de materiais deve incluir também, em referência a cada material, informações sobre o tratamento térmico apropriado e seus parâmetros de especificação. Deve oferecer diferentes possibilidades de pesquisa para facilitar aos projetistas a escolha do material. Por exemplo: pesquisa de acordo com o nome do material, código do material, propriedades desejadas no material, etc.

Um segundo tipo de atributos tecnológicos gerais é representado pelas tolerâncias gerais. A aplicação das tolerâncias gerais, que são aceitas na prática e apoiadas pelas normas DIN, foram consideradas no projeto. As aplicações destas tolerâncias pelo usuário forneceram as informações necessárias para a implementação do protótipo. É importante mencionar, também, que a utilização de tolerâncias geral facilita as atividades dos projetistas e permite ter um modelo de peça mais compacto.

As tolerâncias gerais compreendem as especificações de tolerâncias dimensionais, geométricas e de posição, como também qualidade superficial. Estas especificações afetam todas as dimensões da peça e *manufacturing features*, como também seus elementos, os quais não possuem uma tolerância dimensional, geométrica ou de posição específica, ou uma especificação de qualidade superficial.

Estas tolerâncias são definidas nas normas ISO, as quais são utilizadas como referência para o projeto, tanto para as tolerâncias gerais para dimensões, chanfros, ângulos, retilineidade, planicidade, cilindricidade, perpendicularidade e simetria, quanto para as tolerâncias gerais para qualidade superficial [64].

O protótipo deve proporcionar uma interface amigável com o usuário para a definição das tolerâncias gerais durante o projeto. As tolerâncias gerais não definem apenas uma restrição para a usinagem da peça e suas *manufacturing features*, mas algumas destas tolerâncias também definem uma interdependência implícita entre *manufacturing features* e seus elementos. Como exemplo pode ser mencionada a especificação de uma tolerância geral de perpendicularidade ou simetria. Estas tolerâncias estão, adicionalmente, impondo uma interdependência, o que não está relacionado a uma *manufacturing feature* específica, mas a todas as faces perpendiculares, ou objetos simétricos.

Portanto, este tipo de tolerância “geral” deve ser representado como uma característica genérica de toda a peça; estes atributos estão relacionados à peça e não a uma *manufacturing feature* específica.

Neste caso, na definição de um tratamento térmico específico para uma *manufacturing feature* ou seus elementos, não existe o caráter substituto como existe na definição de uma tolerância geral e uma tolerância específica para uma *manufacturing feature*. A definição do tratamento térmico para uma *manufacturing feature* específica tem um caráter complementar, que deve ser considerado e deve ser coerente com a definição do tratamento térmico geral válido para toda a peça. Por exemplo, a definição de uma têmpera para as faces laterais de um *slot*, que será utilizado como guia e o tratamento térmico geral da peça.

Os projetistas têm, para a especificação do tratamento térmico apropriado, a ajuda de uma interface interativa e a informação disponível no banco de dados de materiais, o que relaciona os materiais com os possíveis tratamentos térmicos. Estas especificações devem considerar também as normas internas específicas da empresa, as quais devem ser definidas através de uma interface com o usuário para o banco de dados de materiais e tratamentos térmicos. Estes tratamentos térmicos gerais, como os outros atributos tecnológicos gerais, também estão relacionados com toda a peça.

4.5.2 Atributos Tecnológicos baseados em *FEATURES*

Os atributos tecnológicos baseados em *manufacturing features* que devem existir no protótipo são as tolerâncias e o tratamento térmico para uma *feature* específica, para um elemento da *feature* ou um grupo deles. As tolerâncias suportadas correspondem às normas internacionais.

Considerando a estrutura lógica do modelador baseado em *features* e a representação das interdependências, o que é fundamental para as atividades do planejamento de processo, os atributos tecnológicos são divididos em dois grupos.

No primeiro estão os atributos tecnológicos, relacionados a um parâmetro ou a um elemento da *feature* de uma única *manufacturing feature;* portanto, esta informação tecnológica (restrição) é pertence a uma *manufacturing feature*.

O segundo grupo, diferente do primeiro, inclui atributos tecnológicos que definem uma restrição entre duas ou mais *manufacturing features* ou elementos de *features* de diferentes *manufacturing features*. Como conseqüência, este grupo de atributos constrói uma interdependência tecnológica entre *manufacturing features*, que devem ser consideradas pela definição de *setups* e seqüências de operações de usinagem durante a tarefa de planejamento de processo. O processo de armazenagem no modelo de informações está descrito no item 5.4.3.

Baseando-se na definição da estrutura lógica acima, é necessário diferenciar uma tolerância dimensional de um parâmetro de uma *manufacturing feature* (tolerância dimensional interna) de uma tolerância dimensional de um parâmetro de posição da *feature* (tolerância dimensional externa), como por exemplo, uma tolerância para a distância entre dois furos. No primeiro caso, a tolerância pertence a uma *manufacturing feature* e no segundo determina uma interdependência entre *manufacturing features,* conforme apresentado em 4.4.1.

4.5.2.1 Atributo Interno da Manufacturing Feature

Considera-se atributo interno da *manufacturing feature* aquele relacionado a um parâmetro ou elemento de apenas uma *manufacturing feature*. Por exemplo, a tolerância dimensional do diâmetro de um furo, ou a tolerância de planicidade das

faces laterais de um s*lot*. Estes atributos tecnológicos são armazenados em uma *manufacturing feature* e determinam uma restrição de manufatura, que é considerada isolada pelo processo de usinagem da *feature* específica. No protótipo, estes atributos serão representados ligados a uma *manufacturing feature*.

Este grupo de atributos tecnológicos inclui tolerância dimensional interna, tolerância geométrica, qualidade superficial e tratamento térmico.

4.5.2.2 Atributo externo da Manufacturing Feature

Os atributos tecnológicos baseados em *features* deste grupo são a tolerância dimensional externa e a tolerância de posição. Estes atributos tecnológicos, em conjunto com uma definição de restrição tecnológica a uma *manufacturing feature*, impõem uma restrição tecnológica entre duas ou mais *features*, o que resulta em interdependências tecnológicas, que devem ser consideradas nas atividades de planejamento de processo.

Estes atributos tecnológicos determinam uma tolerância para um elemento de uma *manufacturing feature* em relação a, pelo menos, um elemento de *manufacturing features* referenciadas.

As interdependências resultantes serão representadas no modelo da peça, relacionando o elemento de uma *manufacturing feature*, o qual adquiriu a tolerância, e os elementos de referência da outra *manufacturing feature*. Estas informações são fundamentais na cadeia de desenvolvimento do processo. São necessárias para o projeto de fixação da peça, a determinação de *setups* e a seqüência de operações de usinagem. Portanto, é fundamental reconhecer estas interdependências e transferi-las de maneira apropriada para os sistemas subseqüentes da cadeia de processo.

4.5.3 Validação de Tolerâncias

Para que a introdução de tolerâncias no modelo da peça seja coerente e válida, o sistema deve possuir um sistema de validação para que as tolerâncias definidas

no modelo possam ser avaliadas, de acordo com o conhecimento disponível de tolerância, no momento em que o projetista estiver modelando a peça.

Um exemplo disso é a definição de uma perpendicularidade, em que o módulo de projeto deve verificar se o elemento que recebeu a tolerância e o elemento de referência são realmente perpendiculares e se o valor da tolerância é coerente aos valores utilizados.

São várias as regras para validação de tolerâncias, dependendo do tipo de tolerância aplicada. Para uma tolerância de cilindricidade, por exemplo, verifica-se, além do valor definido, se a face que está recebendo a tolerância é realmente cilíndrica.

5 Implementação

A implementação do modelador baseado em *manufacturing features* utilizado no projeto *FESTEVAL,* se deu através da criação de uma biblioteca de vínculos dinâmicos, ou seja, a funcionalidade dada pelo *FESTEVAL* ao modelador proprietário da Unigraphics está armazenada em um arquivo compilado que contém todas as funções implementadas.

Este projeto de pesquisa foi concluído com a implementação das *features*: *chamfer, hole, planar face, pocket, slot* e *thread,* e dos atributos tecnológicos de tolerância dimensional, de forma, e de posição. A possibilidade de armazenar o modelo de acordo com as normas STEP foi implementado enquanto que a geração do modelo através de um arquivo não foi desenvolvida.

A criação desta biblioteca foi feita com o uso do software Microsoft Visual C++, dentro do paradigma da orientação a objetos. Neste ambiente de desenvolvimento, foi criada uma área de trabalho que inclui quatro projetos de software integrados. Este projetos de software constituíram a biblioteca de vínculos dinâmicos, como apresentado na Figura 5.1.

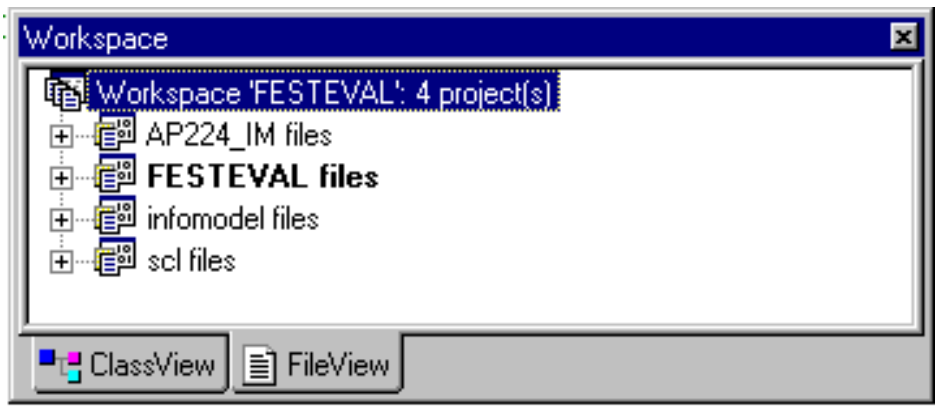


Figura 5.1: Área de trabalho para o desenvolvimento do projeto

Os projetos de softwares utilizados na biblioteca Festeval.dll foram:

- *FESTEVAL*;
- AP224_IM;

- SCL;
- Infomodel.

O projeto de software ***FESTEVAL*** contém a funcionalidade para elaboração de um modelo baseado em *features.* Nesta dissertação, será tratado o desenvolvimento deste projeto de software.

O projeto de software **AP224_IM** foi desenvolvido pelo DiK e define o modelo de dados do produto em linguagem de programação C++, de acordo com o protocolo de aplicação AP224 das normas STEP [61].

O projeto de software **SCL** é uma biblioteca de classes desenvolvida pelo NIST - *National Institute of Standards and Technology* para o desenvolvimento de software com as normas STEP, esta biblioteca tem o objetivo de facilitar o uso industrial das normas STEP e é disponibilizada sem custos [65].

O projeto de software **Infomodel** consiste em um conjunto de classes que dão funcionalidade à transferência dos dados do modelo para o arquivo STEP e resulta de trabalhos do PTW e DiK [61].

As ferramentas CASE existentes podem gerar o código-fonte diretamente dos diagramas de classe, bem como montar diagramas de classe de um código-fonte existente. As duas possibilidades foram utilizadas no desenvolvimento do *FESTEVAL*, utilizando o software Rational Rose.

5.1 Interface com o usuário

Atualmente, o desenvolvimento de novos projetos na indústria deve ser realizado no menor tempo possível e da maneira mais fácil que se possa fazer. Apesar de os sistemas CAD terem ajudado muito neste aspecto, é necessário que estes sistemas tenham uma interface amigável com o usuário, permitindo rápido acesso às informações e fácil construção de novos modelos.

Para a implementação do menu do modelador baseado em *manufacturing features*, é necessário um estudo das necessidades do usuário e das funcionalidades necessárias ao projeto.

Esta interface com o usuário foi estruturada e dividida em partes que permitem um acesso lógico e simples às funcionalidades do protótipo (

Figura 5.2). Para isso, várias opções foram estudadas e implementadas, até ser encontrada uma estrutura final. Estas implementações foram analisadas pelos usuários finais, representados pelas Indústrias Romi S. A., e modificadas de acordo com suas necessidades.

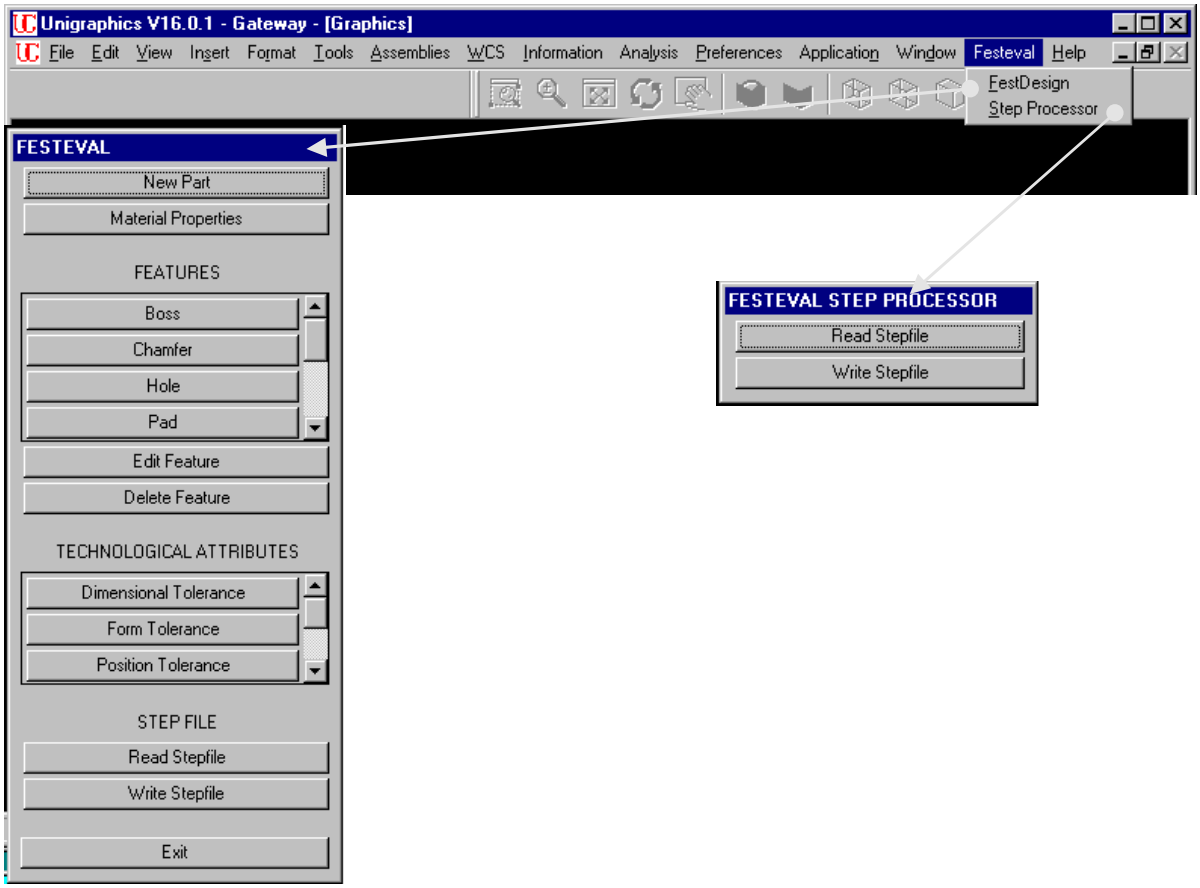


Figura 5.2: Interface com o usuário

A estrutura do menu do modelador foi definida baseada no menu do próprio Unigraphics, em que o usuário escolhe uma aplicação e todas as funcionalidades disponíveis para aquela aplicação são mostradas em uma única janela (Figura 5.3). O usuário escolhe a aplicação "*Festeval*" e todos os métodos disponíveis são apresentados em uma janela para o projetista.

O menu inicial *Festeval* é dividido em duas partes:

- "*FestDesign*": oferece todas as funcionalidades para a modelagem de peças através de *features*, incluindo a importação e exportação do modelo de informações do *FESTEVAL* através do arquivo físico STEP;

- "*STEP Processor*": permite o acesso direto às funções STEP implementadas no protótipo *FESTDESIGN*. Este item foi criado para melhorar o acesso do usuário quando é necessário apenas ler ou escrever arquivos STEP. É dividido em "*Read STEP File*", para ler um arquivo STEP, e "*Write STEP File*", para gerar um arquivo STEP.

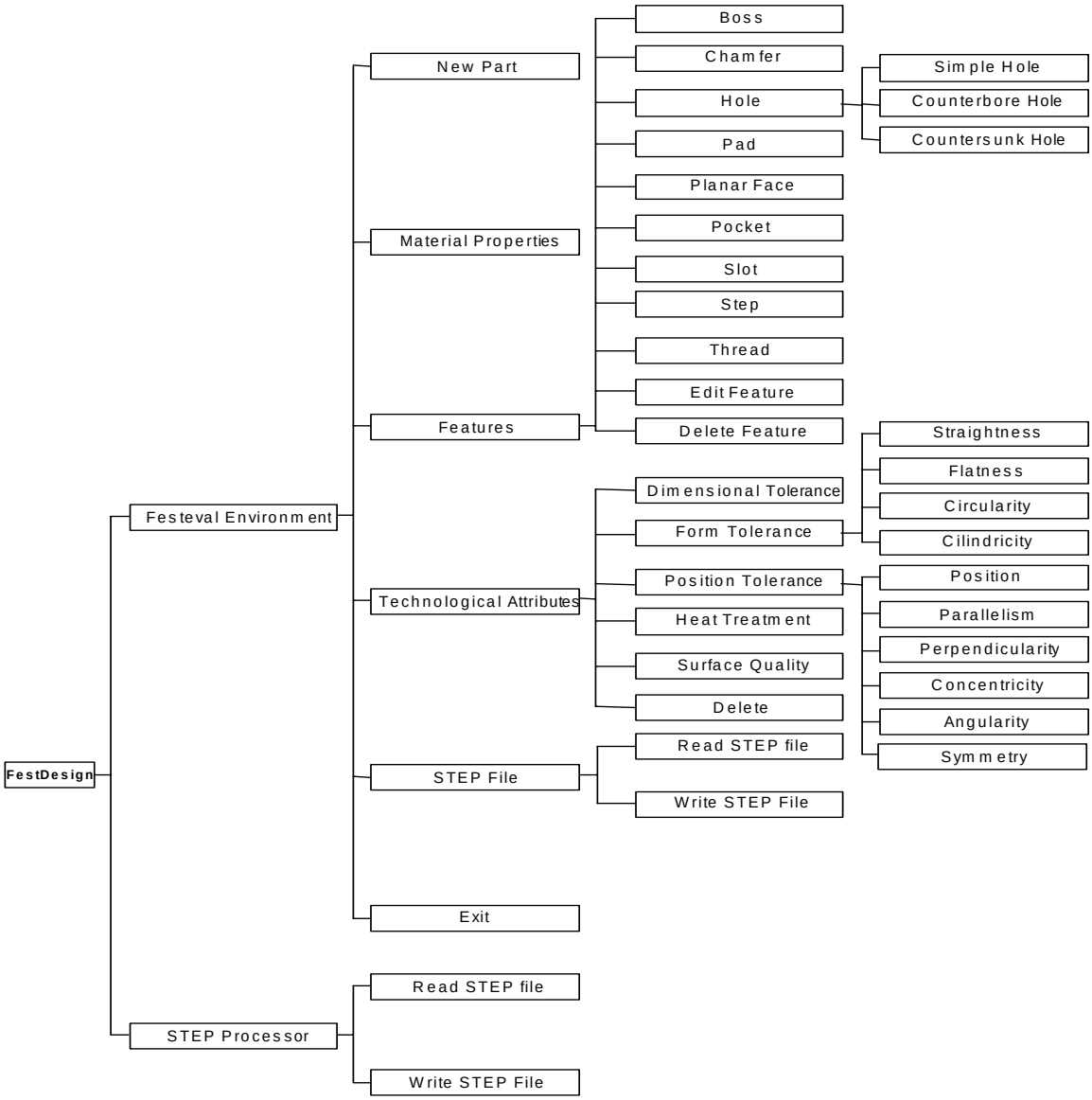


Figura 5.3: Estrutura do menu do FestDesign.

O menu "*FestDesign Environment*" é dividido em cinco categorias, agrupando funções e operações similares: definição do *blank* (*New part*), entrada de atributos gerais (*Material Properties*) modelagem baseada em *features* (*Features*), entrada de atributos tecnológicos (*Technological Attributes*) e geração da base de dados STEP (*STEP File*). Neste menu também está a opção de finalizar o protótipo (*Exit*).

O comando *New Part* inicia a aplicação. Ele deve ser utilizado para introduzir uma peça que será utilizada como um *blank* no ambiente de projeto. Se esta peça também foi modelada utilizando o sistema Unigraphics e inclui algumas *features* geométricas do Unigraphics, o módulo *FESTDESIGN* pode ser utilizado para reconhecer estas *features* na peça (*blank*). Um simples *blank*, como um bloco de material, também pode ser utilizado pelo protótipo.

O menu *Material Properties* permite a escolha do material da peça que está sendo modelada. Isto permite que o usuário defina as propriedades do material a ser utilizado no processo de manufatura, para obter os parâmetros de corte e outras informações relacionadas ao material.

O item *Features* possibilita a utilização de algumas *features* na modelagem da peça, escolhidas de acordo com as necessidades do usuário. Neste caso, foram escolhidas peças de fabricação comuns nas Indústrias Romi S. A. e avaliadas as *features* que seriam necessárias para a modelagem destas peças. Adicionalmente, existem os itens *Edit Feature*, para possibilitar a edição de *features* já construídas, e *Delete Feature*, para remover uma *feature* modelada anteriormente.

O menu *Technological Attributes* permite a introdução de atributos tecnológicos baseados em *manufacturing features* no modelo da peça. Estes atributos incluem: tolerâncias, tratamento térmico e qualidade superficial. As tolerâncias foram divididas em tolerância dimensional, geométrica e de posição, e podem ser aplicadas em todas as *features* que pertencem ao modelo de informações do *FESTEVAL*.

O item STEP *File* é dividido em duas partes: *Write STEP File,* que permite a exportação de um arquivo físico STEP baseado no modelo de informações do *FESTEVAL*, definido de acordo com as normas STEP AP 224 [41], e *Read STEP*

File, que, posto que não implementado, será o caminho para a importação de um arquivo STEP gerado anteriormente, possibilitando a geração de um modelo baseado em *features* através da leitura de um arquivo físico STEP. O arquivo físico STEP auxilia a integração digital com os módulos do *FESTEVAL*: *FESTPLAN*, *FESTNC* e *FESTCONTROL*.

O menu *Exit* encerra o processo da peça em desenvolvimento. Depois disso, todas as funções devem ser bloqueadas e o usuário deve reiniciar uma nova sessão para utilizar o sistema novamente.

5.2 Banco de Dados

Para que não haja redundância nas informações armazenadas do modelo criado é utilizado somente um banco de dados, como é estritamente necessário o uso do banco de dados do UG pelo uso deste software na modelagem foi escolhida a opção de estender a capacidade deste banco de dados para atender às necessidades do projeto. Nesta solução o banco de dados no formato STEP somente é criado no momento da criação do arquivo físico STEP e é imediatamente apagado, como mostra a Figura 5.4.

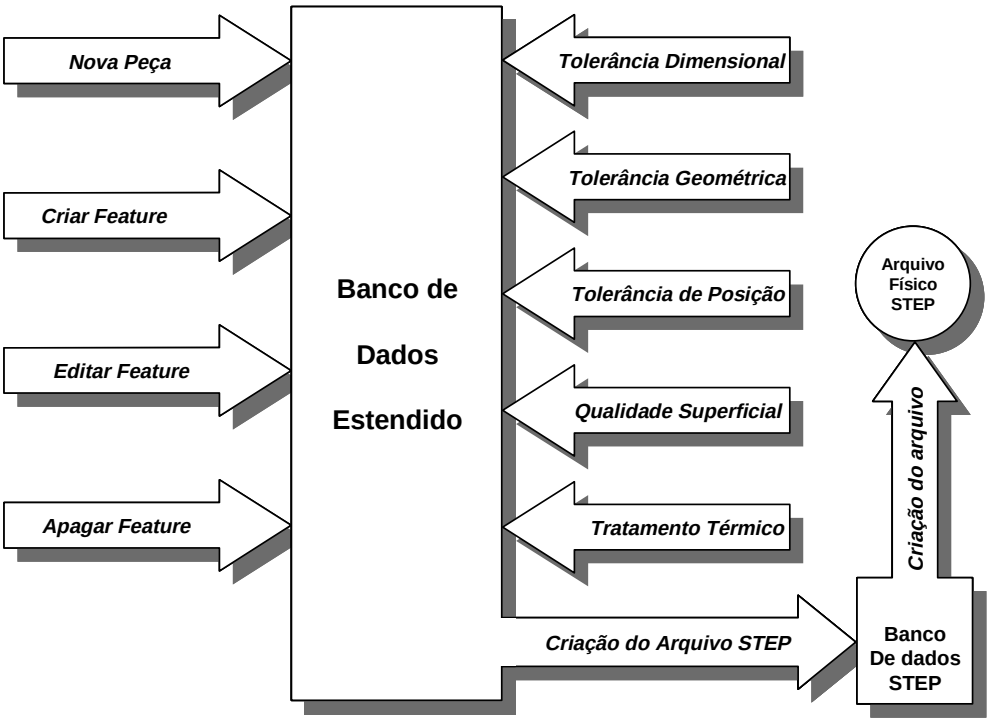


Figura 5.4: Armazenamento em um único bando de dados [51].

Uma vantagem desta solução é a capacidade do software utilizado agregar atributos a qualquer elemento armazenado.

5.3 Estrutura Geral do Programa

A estrutura estática do programa desenvolvido no projeto de software *FESTEVAL* para o modelador baseado em *features* é derivada da informação contida no modelo de referência de aplicação ARM do protocolo de aplicação AP224 modificado (item 4.2) das normas STEP, como pode ser observado na Figura 5.5.

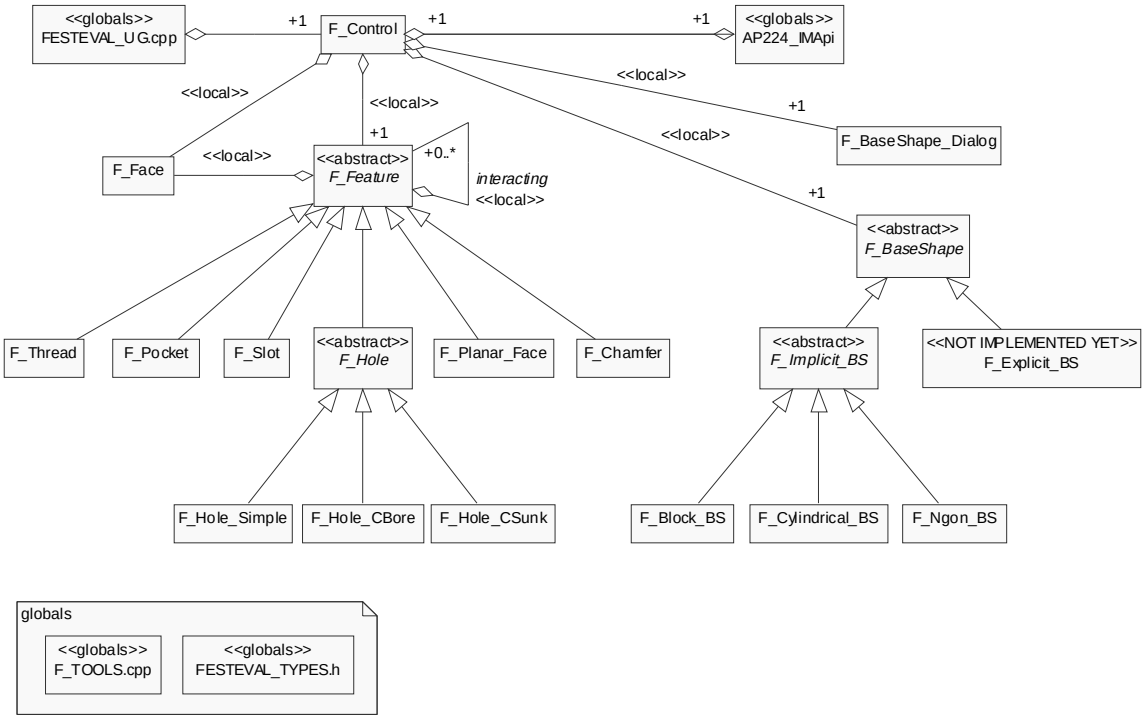


Figura 5.5: Diagrama de classes geral simplificado

As três classes do programa desenvolvido que dão funcionalidade ao programa são:

- F_Feature;

  É uma superclasse virtual que contém os métodos e atributos comuns a todas as classes que servem de modelo para a criação dos objetos que manejam os dados e funções das *features*.

- F_Control;

  É a classe que contém os métodos que dão funcionalidade à interface com o usuário, basicamente fazendo o controle das caixas de diálogo.

  Seus métodos dão funcionalidade para a criação de uma nova peça, para a inserção e remoção de uma *feature*, para a criação do arquivo STEP e para a inserção de atributos tecnológicos.

- F_Base_Shape.

  É uma superclasse virtual para o bloco que receberá as *features* criadas.

5.4 Métodos Gerais das subclasses de F_Feature

A classe abstrata F_Feature serve de modelo para todas as classes das *manufacturing features* (veja Figura 5.6).

Os métodos da classe F_Feature foram criados com a idéia de dar funcionalidade ao que é comum a todas as *features*. Seus atributos também pertencem, por herança, a todas as classes das *features*.

Para um melhor entendimento dos métodos desta classe, é possível dividi-los em:

- Métodos para criação, edição e remoção de *features;*
- Métodos para validação
- Métodos para inserção de atributos
- Métodos para obtenção de informações sobre as *features*

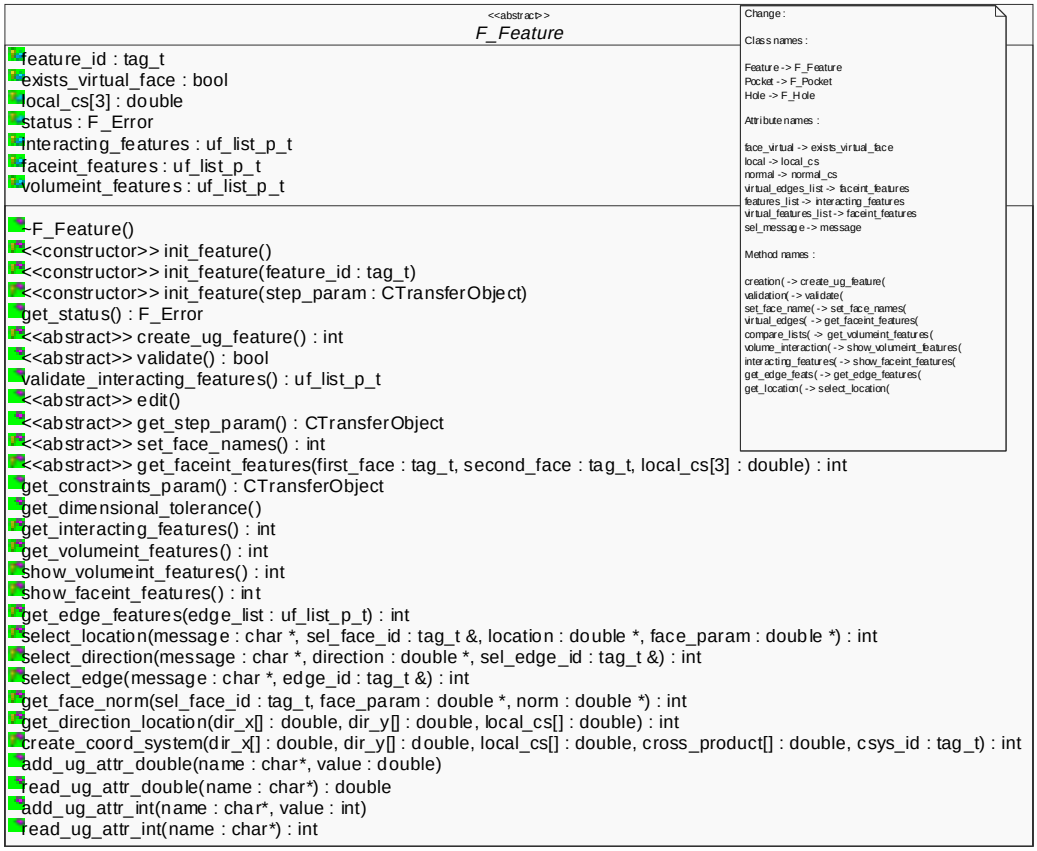


Figura 5.6: Diagrama de classe para F_Feature

5.4.1 Métodos para criação, edição e remoção de *features*

A instanciação de uma *manufacturing feature* no modelo é feita de acordo com os procedimentos descritos no diagrama de casos de uso da Figura 5.7., e obedece ao diagrama de seqüência apresentado na Figura 5.8.

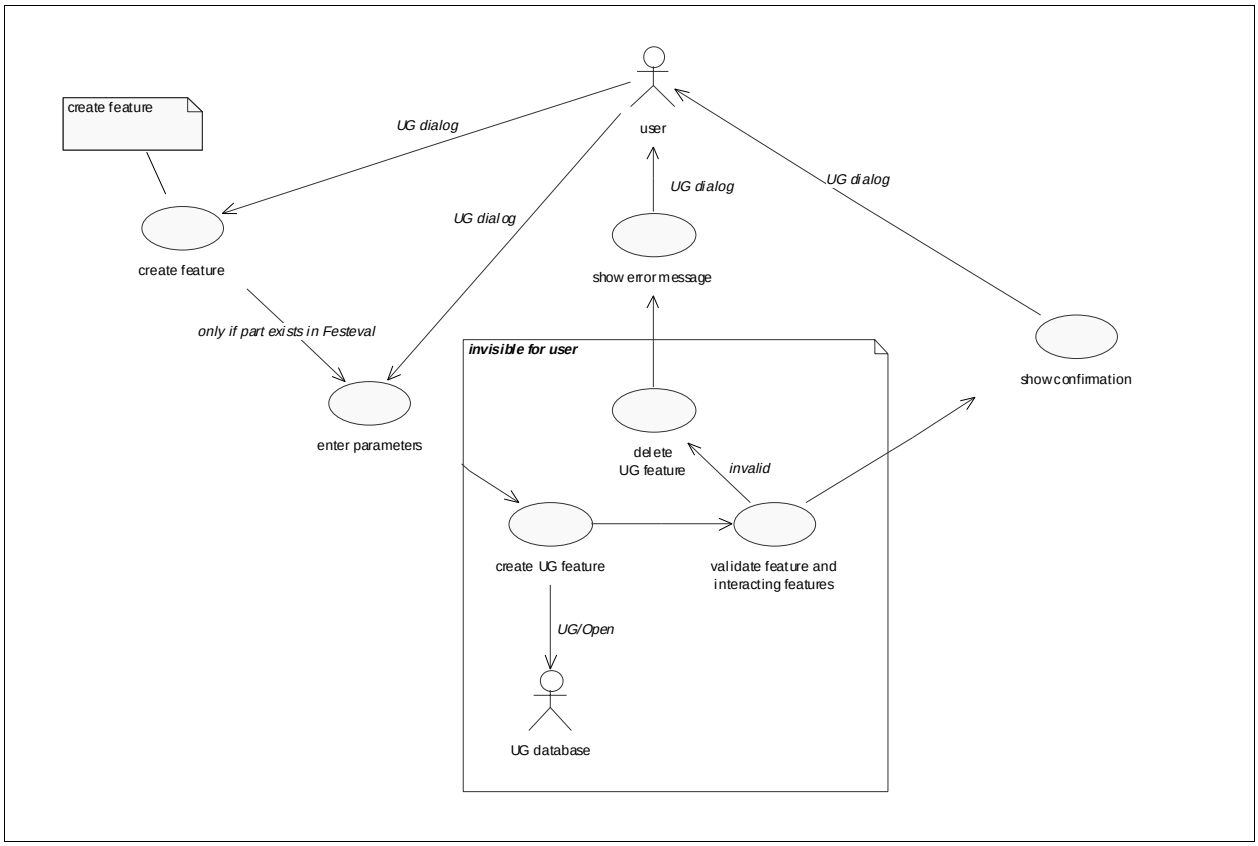


Figura 5.7: Diagrama de caso de uso para a instanciação de uma feature ao modelo

Para a instanciação de uma *feature* são definidos três métodos sobrecarregados, nomeados como **init_feature** que podem ter como parâmetros:

- Um identificador de *manufacturing feature* (feature_id), para a transformação de uma entidade do Unigraphics em *manufacturing feature*;
- Um parâmetro STEP (CStepParam), para a recuperação da informação existente em um arquivo STEP;
- Um parâmetro vazio (*void*) para a construção de uma nova *feature*;

O primeiro procedimento, antes da instanciação da *feature,* é a marcação do instante anterior à instanciação, para que, se a *feature* criada não for válida, possa haver o retorno a esta posição, sem alteração do modelo.

Para a instanciação da *feature,* o método de F_Feature chama o método específico para cada *feature*, que é o método virtual “create_ug_feature”.

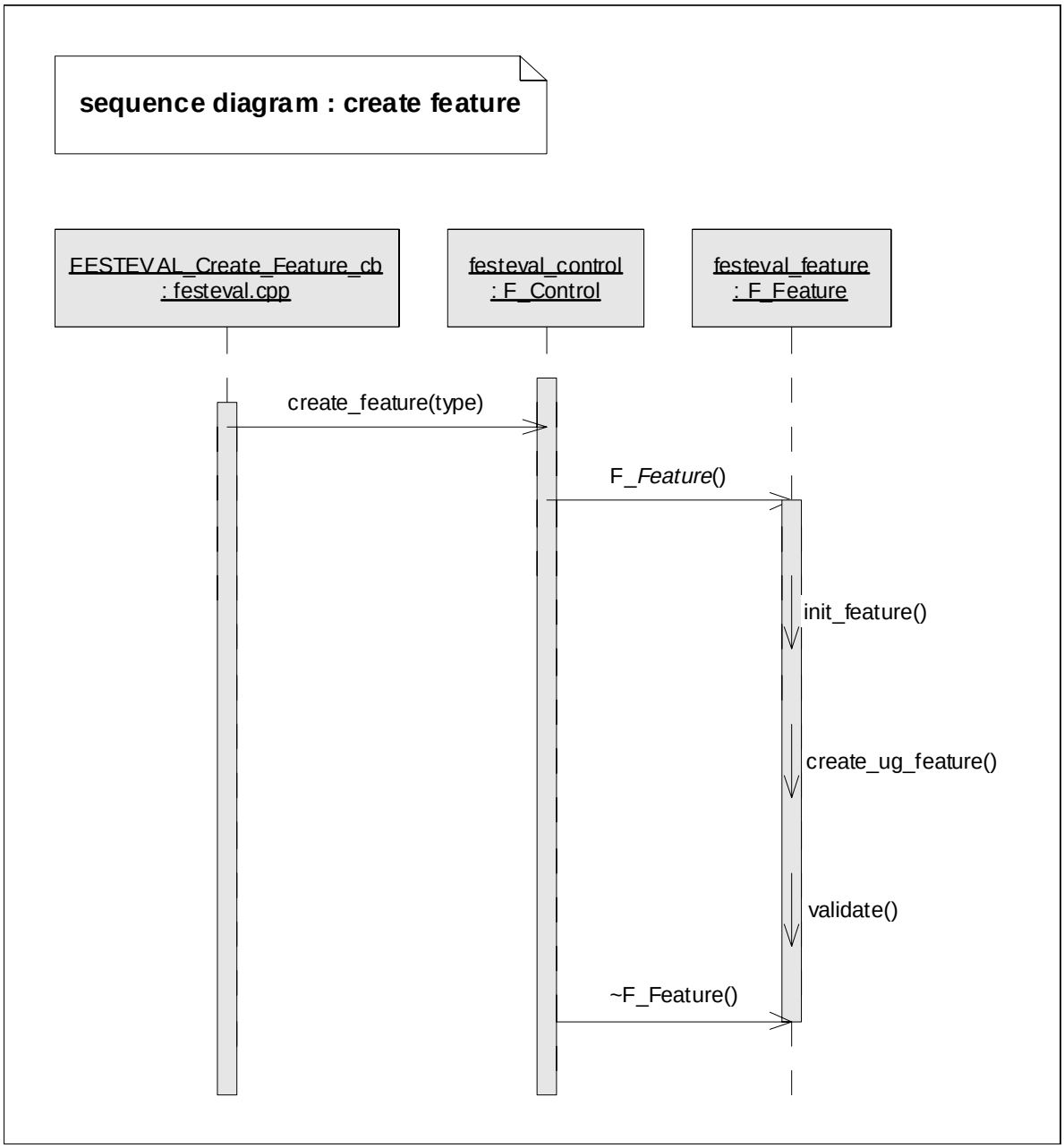


Figura 5.8: Diagrama de seqüência para a inserção de uma feature ao modelo.

Após a criação da *feature*, são efetuados os seguintes procedimentos: a validação da *feature* em si; a nomeação das faces e em seguida a validação de todas as *features* que interagem com a *feature* criada. A criação da *feature* somente é concretizada no caso da mesma ser válida e se a criação não invalidar nenhuma das *features* que com ela interajam, neste trabalho serão apresentados os métodos para a *manufacturing feature pocket.*

O diagrama de estrutura da Figura 5.9 demonstra o funcionamento do método para criação de uma *feature*.

Para indicar ao usuário o resultado da criação da *feature* são apresentadas duas caixas de diálogo, uma para a correta inserção da *feature*, apresentada na Figura 5.10, e outra indicando que a *feature* escolhida não pode ser inserida, conforme Figura 5.11.

Na criação da *feature,* também é executado o método que gera um sistema de coordenadas locais.

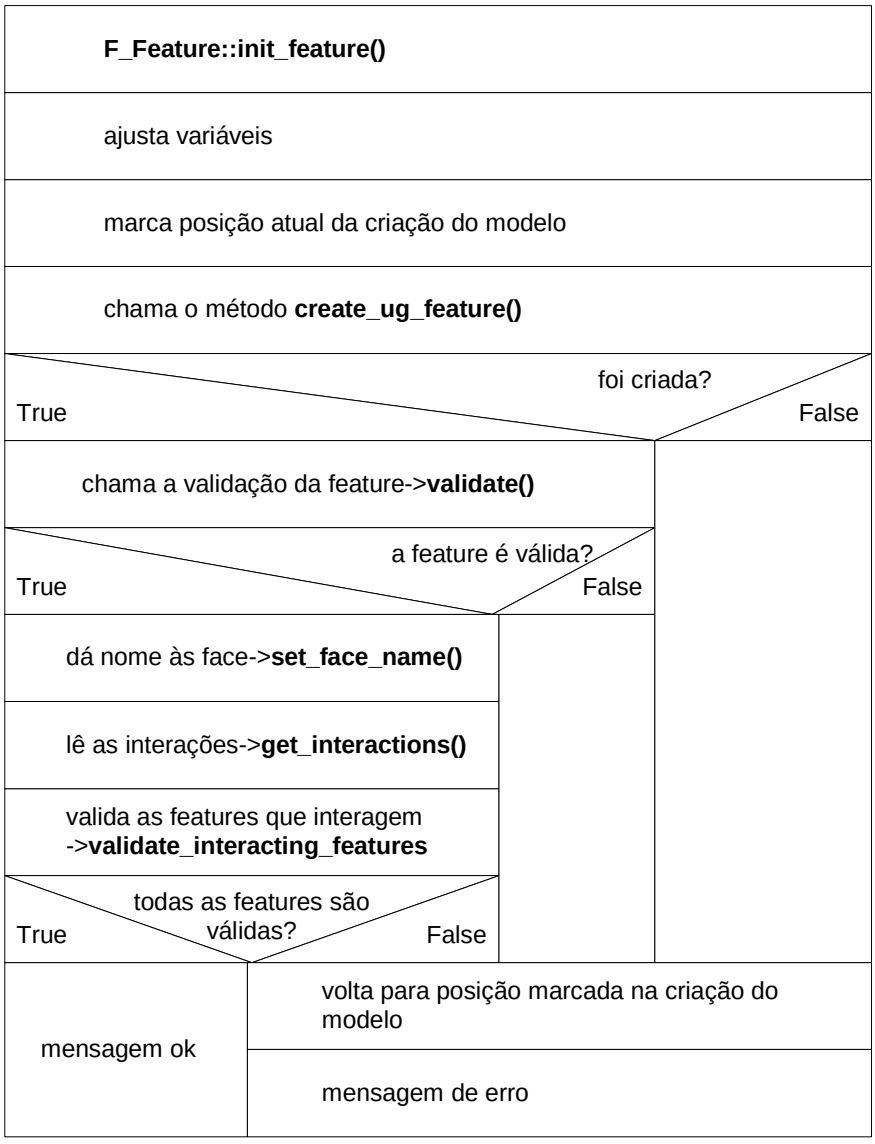


Figura 5.9: Diagrama de estrutura do método para criação de uma feature.

mensagem ok
uc1601("Feature is valid and was created successfully !", 1);

Figura 5.10: Caixa de diálogo indicando a correta criação da feature.

mensagem de erro
uc1601("Feature can not be done.",1);

Figura 5.11: Caixa de diálogo informando que a feature desejada não pode ser criada no FESTEVAL

5.4.2 Métodos para obtenção de informações sobre as *manufacturing features*

Os métodos para obtenção de informações sobre as *manufacturing features* estão divididos em métodos que recuperam informações e os que escolhem parâmetros. Ambos dão apoio aos outros métodos que utilizam estas informações, seja para a instanciação de novas *features* ou para as transferências das informações para o banco de dados.

Os métodos de obtenção de informações atualizam as informações do sistema de coordenadas e do vetor normal à *feature*. Os métodos que escolhem parâmetros são utilizados na criação das *features* e servem para a escolha de faces e arestas como referências para a criação de *features*.

Como exemplo, pode ser citado o método “select_location”, utilizado para determinar a face de criação de uma *feature*. Ao se iniciar este método, é pedido ao usuário a escolha de uma face do modelo, através do método “select_face”. Com a face escolhida, são lidos seus parâmetros que retornam para a função de criação da *feature.* Na Figura 5.12, são apresentados os métodos e as caixas de diálogo para estes métodos. No modo de operação natural do software CAD, o ponto de referência para a criação da *feature* é o ponto em que houve a escolha da face pelo mouse. Com o projeto *FESTEVAL,* houve uma melhoria nesta interface, pois o ponto de referência passou a ser sempre o centro da face escolhida.

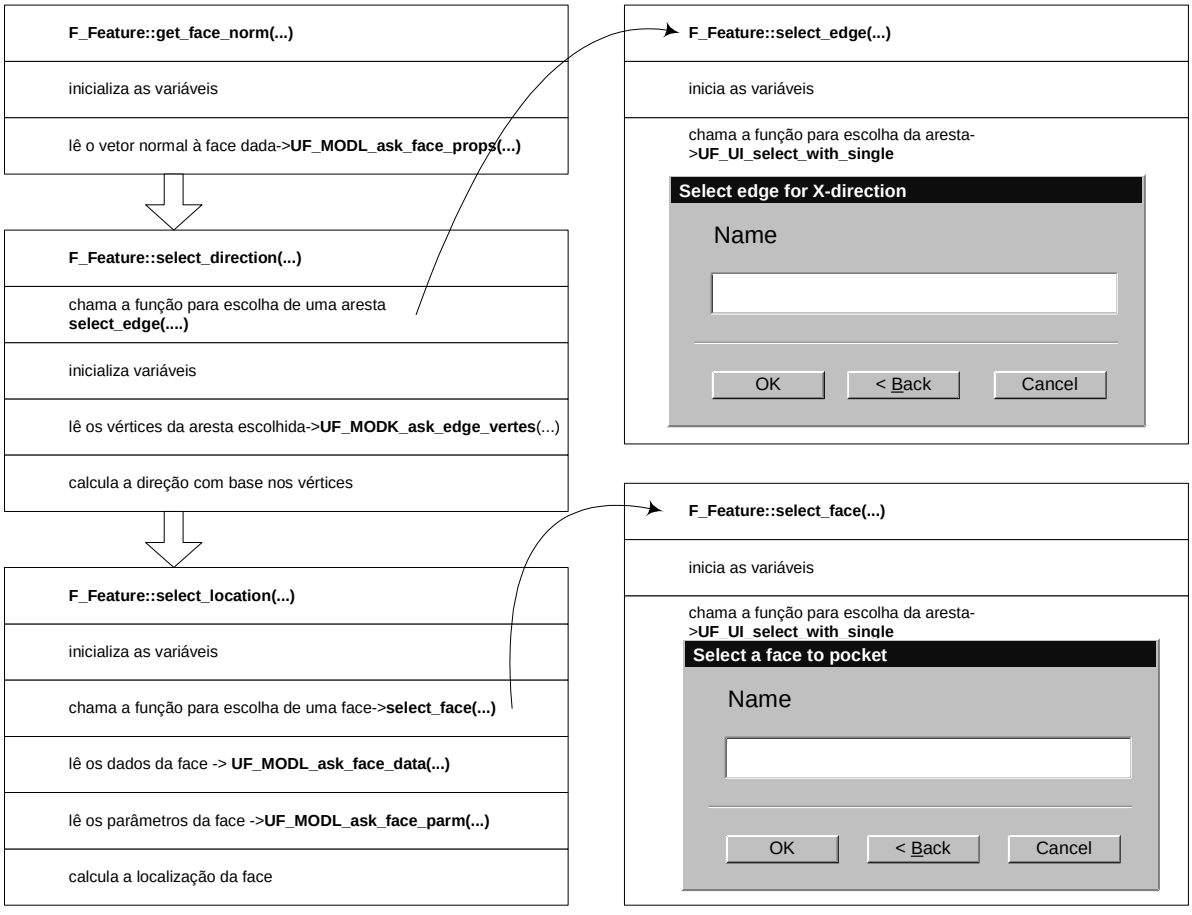


Figura 5.12: Métodos de obtenção de informações das features.

5.4.3 Métodos para inserção de atributos

Os métodos para a inserção de atributos colocam, no banco de dados estendido [4.4.1], as informações tecnológicas referentes à *feature*. Atributos de forma geral podem ser lidos ou vinculados à uma *feature* na forma de um número real ou inteiro [66], através dos métodos apresentados na Figura 5.13.

```
add_ug_attr_double(char* name, double value);
add_ug_attr_int(char* name, int value);
read_ug_attr_double(char* name);
read_ug_attr_int(char* name);
```

Figura 5.13: Métodos para inserção de atributos

Atributos são inseridos no banco de dados estendido do Unigraphics, como se fossem envelopes em que é escrito o destinatário como texto e o conteúdo como um valor numérico. A recuperação da informação se dá com a função de leitura em que o destinatário é o parâmetro.

Os atributos tecnológicos são inseridos por meio de métodos pertencentes à superclasse F_Feature; cada um dos atributos tem seu método específico.

Como exemplo, pode ser apresentado o método de inserção do atributo tecnológico correspondente à tolerância de cilindricidade, como demonstrado nos diagramas de estrutura da Figura 5.14.

O método só é iniciado após a escolha da *manufacturing feature* à qual estará vinculado o atributo tecnológico. Após a inicialização das variáveis, é apresentada uma caixa de diálogo para a escolha face que receberá o atributo de tolerância de cilindricidade. A esta face é feita a verificação: se é uma face cilíndrica e se pertence à *feature* escolhida. Confirmadas estas verificações, é lido o atributo de cilindricidade previamente inserido, se houver. Finalmente é apresentada uma caixa de diálogo para a inserção do valor vinculado à tolerância de cilindricidade. Uma melhoria implantada no programa é a apresentação na caixa de diálogo do valor anteriormente inserido.

A inserção no banco de dados estendido se dá através da inclusão de um atributo vinculado à *feature* que tem como nome o texto “CYLINDRICITY” e, como valor numérico, a tolerância inserida.

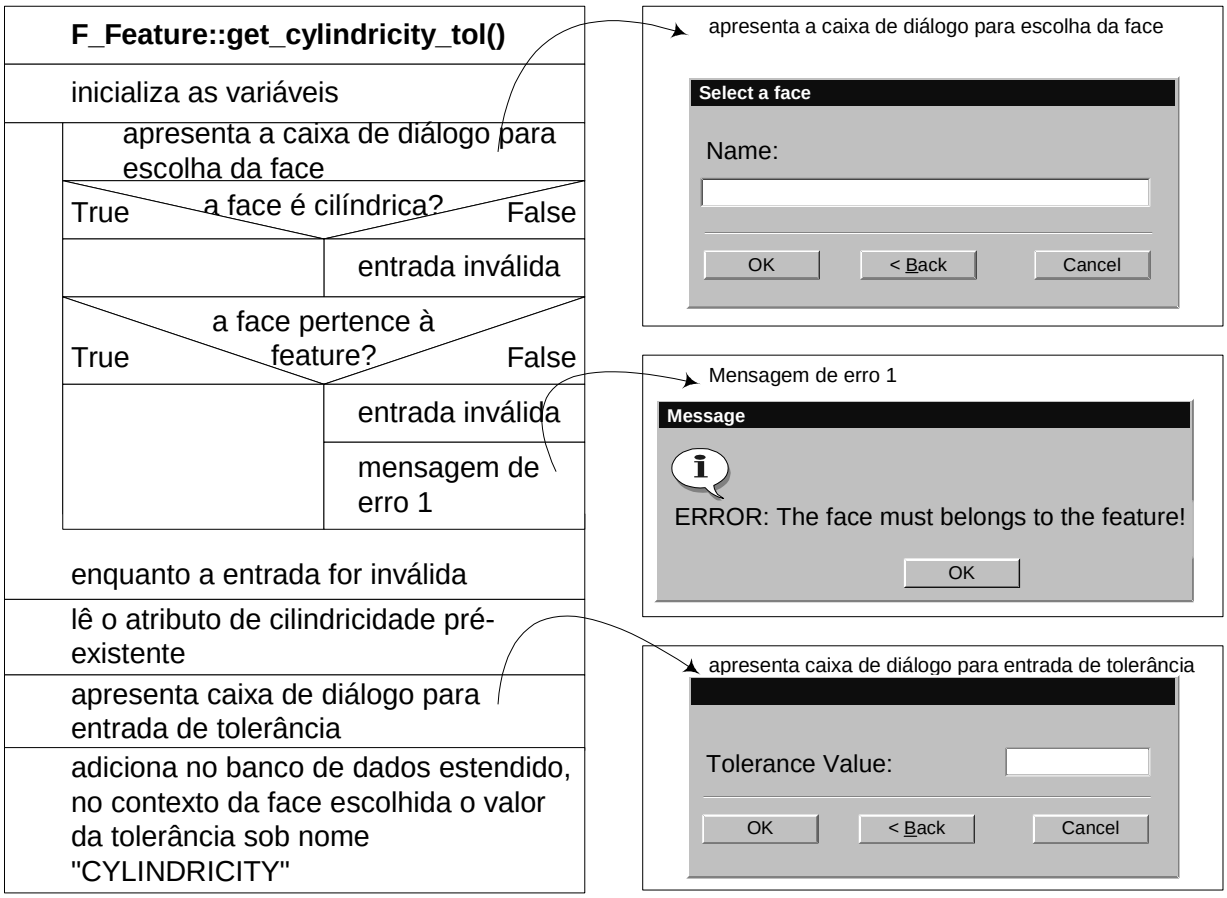


Figura 5.14: Diagramas de estrutura e caixas de diálogo para o método "get_cylindricity_tol"

5.4.4 Atributos da Classe F_FEATURE

A superclasse F_Feature define atributos que são transmitidos por herança a todas as *features,* conforme apresentado na Figura 5.15. Estes atributos têm caráter geral e são necessários para a funcionalidade de qualquer *feature*. Atributos específicos para cada *feature* são definidos em suas classes.

```
bool exists_virtual_face;
double local_cs[3];
tag_t feature_id;
uf_list_p_t faceint_features;
uf_list_p_t interacting_features;
uf_list_p_t volumeint_features;
```

Figura 5.15: Atributos da classe F_Feature

A existência, de pelo menos uma face virtual, é uma das condições indicativas da validade de uma *feature*, pois é necessária a existência do acesso para a ferramenta. Pela grande importância deste indicativo, foi decidido deixá-lo como atributo geral para todas as *features*, nomeado como **exists_virtual_face** do tipo lógico.

O sistema de coordenadas locais é usado em várias operações para qualquer *feature*; este atributo é definido com o nome **local_cs,** como uma matriz de três dimensões que indicam a posição do sistema de coordenadas locais em relação ao sistema de coordenadas globais. A atualização deste atributo é feita pelo método **create_coord_system.**

O atributo **feature_id** é o identificador da *feature*; é definido com o tipo "tag_t" que é uma variável inteira positiva. O tipo "tag_t" está definido no arquivo "uf_defs.h", que contém as macros e definições para todas as estruturas do Unigraphics, sendo que, especialmente o tipo "tag_t" identifica qualquer entidade do UG, como: *features*, retas, planos ou pontos.

Os três últimos atributos armazenam listas com as *features* que interagem de modo global com a *feature* corrente em **interacting_*features***, com as *features* que interagem por volume em **volumeint_*features*** e com as *features* que interagem por face em **faceint_*features***.

Estes atributos utilizam o tipo lista "uf_list_p_t", que é um ponteiro para uma lista estruturada de entidades do tipo "uf_list_s" (veja Figura 5.16). Esta lista contém, em cada registro, uma entidade e um ponteiro para a próxima entidade e está assim declarada:

```
struct uf_list_s
{
     tag_t eid;
     struct uf_list_s *next;
}
typedef struct uf_list_s *uf_list_p_t;
```

Figura 5.16: Estrutura de dados para entidades do UG

A atualização destas listas é feita através do método "get_interacting_*features*" definido em F_Feature, cujo diagrama de estrutura é apresentado na Figura 5.17.

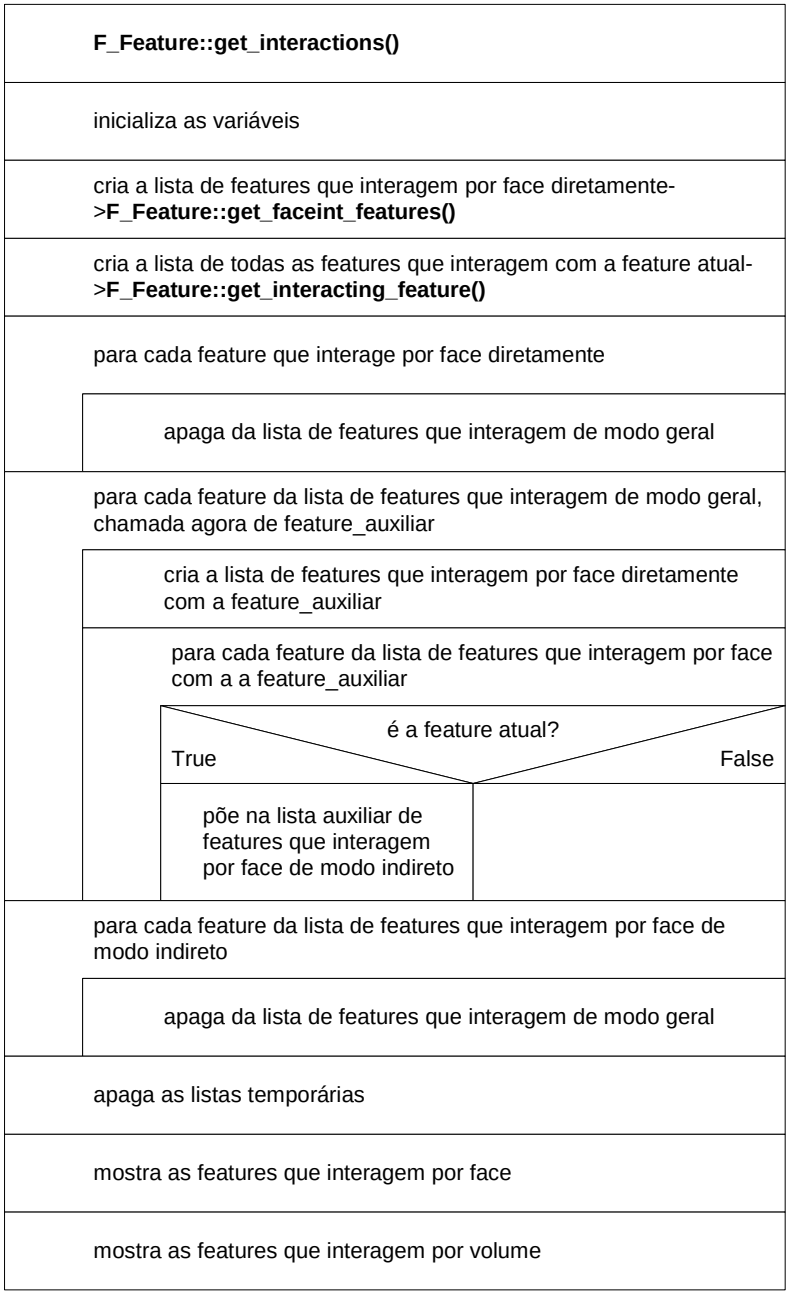


Figura 5.17: Diagrama de estrutura para a obtenção das interações entre features

5.5 Métodos específicos para uma feature pocket

Para demonstrar a implementação, foi escolhida a apresentação dos métodos específicos para um *pocket.*

A classe F_Pocket é derivada da superclasse F_Feature e contém os métodos e atributos pertinentes à manipulação dos objetos *pockets* que representam a

feature pocket. O diagrama de classe para F_Pocket é apresentado na Figura 5.18.

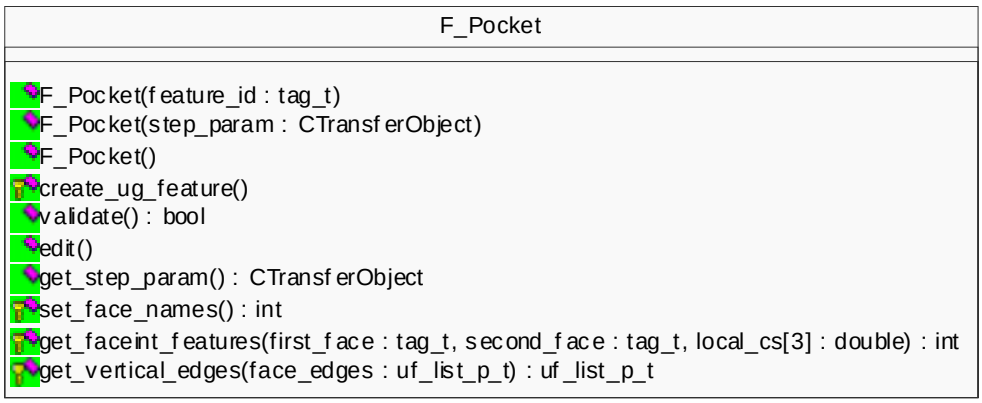


Figura 5.18: Diagrama de classe para F_Pocket

A classe F_Pocket é um molde para a instanciação de um objeto que agrega todos os métodos necessários à criação, validação, edição e transferência para um arquivo STEP de uma *feature pocket*. Seus métodos e atributos são apresentados na Figura 5.19.

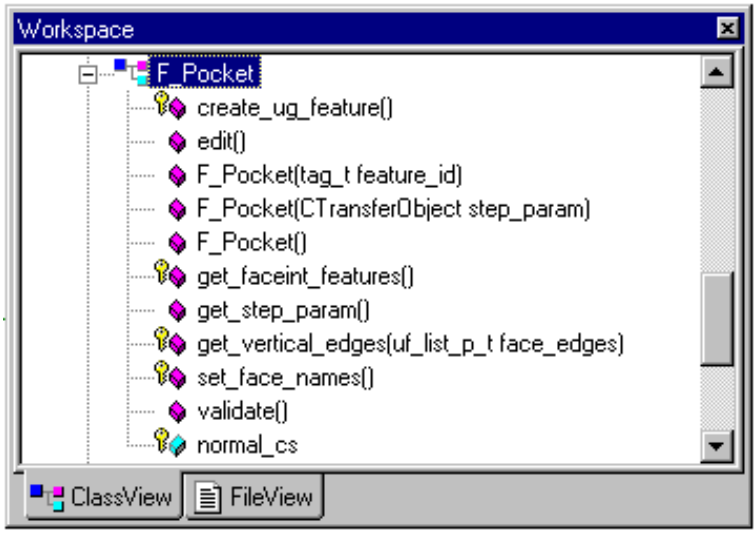


Figura 5.19: Métodos e atributos da classe F_Pocket

Os métodos específicos para uma *feature* dão funcionalidade para três ações distintas, a criação, a edição e a transferência dos dados desta *feature* para o banco de dados.

Os métodos utilizados para a criação de um *pocket* podem ser descritos seqüencialmente com segue:

- ***create_ug_feature()*:** Insere no modelo uma *feature*, com os padrões do UG;
- ***F_Pocket()*:** insere um *pocket* no modelo;
- **validate():** Valida o pocket;
- ***set_face_name()*:** Dá nome as faces do *pocket*;
- ***get_interactions()*:** Obtém as *features* que interagem com o *pocket*;
- ***validate_interacting_features()*:** Valida as features que interagem com o *pocket*;

Para a edição de um pocket é utilizado o método:

- **edit();**

Para a transferência dos dados para o banco de dados são utilizados os métodos:

- **get_step_param():** armazena os dados do *pocket* para transferência para o arquivo STEP;
- **get_faceint_features():** obtém as features que interagem por face com o pocket, é chamado pelo método get_step_param();
- **get_vertical_edges():** obtém do pocket as arestas verticais, é chamado pelo método get_faceint_features().

Os métodos em negrito são métodos virtuais da superclasse F_Feature, descritos na classe F_Pocket e são apresentados a seguir.

5.5.1 Métodos Construtores

Os construtores “F_Pocket” são sobrecarregados e executam a criação da *feature pocket* em três situações distintas:

- **F_Pocket ():** Em uma peça existente, é inserido um *pocket* pelo usuário;
- **F_Pocket (tag_t *feature*_id):** Ao abrir uma peça no ambiente *FESTEVAL*, um *pocket* é validado e inserido no conjunto de *features* do ambiente;
- **F_Pocket (CTransferObject step_param):** Ao ser feita a leitura de um arquivo STEP, é inserido um *pocket* à peça. Este método ainda não foi implementado no protótipo.

5.5.2 Método para Validação da *Feature Pocket*

```
Bool F_Pocket::validate();
```

A validação geométrica e semântica de uma *feature* é um dos métodos mais importantes para o conceito de projeto baseado em *features .* A cada inserção de uma *feature,* é feita a validação da *feature* em si, bem como a validação de todas as *features* que com ela interagem.

O método para validação é definido como um método virtual para a superclasse abstrata “F_Feature”, e sua implementação é feita em todas *features* descendentes.

O retorno deste método é uma variável lógica que indica uma *feature* inválida com o valor falso e uma *feature* válida com o valor verdadeiro.

Para a *feature pocket*, a validação é feita através de duas verificações, conforme discutido em [4.3]:

- Existência de pelo menos cinco faces cilíndricas;
- Possibilidade de um off-set mínimo para todas as faces dentro do corpo.

Para a implementação deste método, foi usado o diagrama de estrutura apresentado na Figura 5.20.

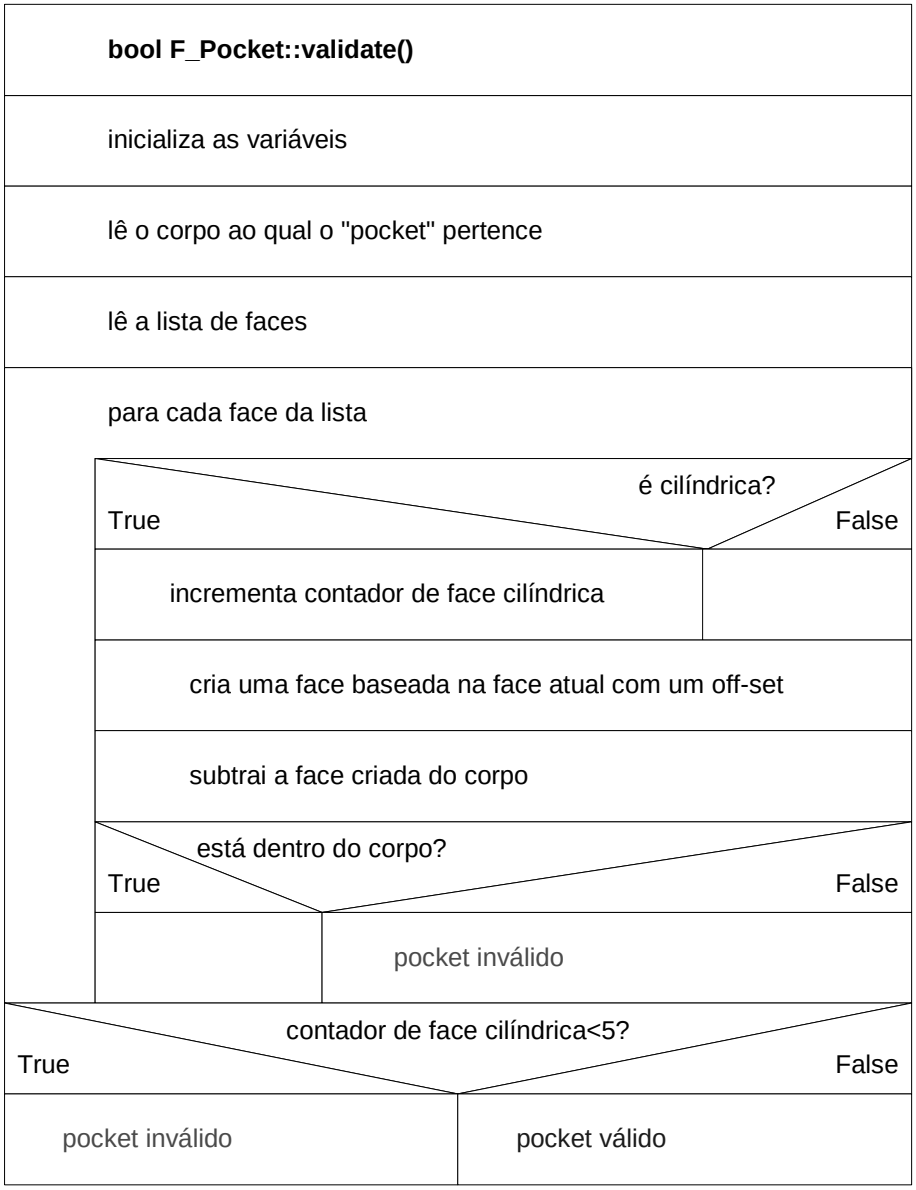


Figura 5.20: Diagrama de estrutura de validação de um rebaixo.

Neste método, a contagem de faces cilíndricas é feita dentro do laço que faz o off-set das faces e se qualquer off-set das faces estiver fora do corpo, o laço é interrompido e o método imediatamente retorna um valor falso.

Na Figura 5.21, são mostradas as faces cilíndricas em vermelho; as faces de raio de fundo são esféricas. Se não houver, pelos menos cinco faces cilíndricas, o *pocket* é considerado inválido.

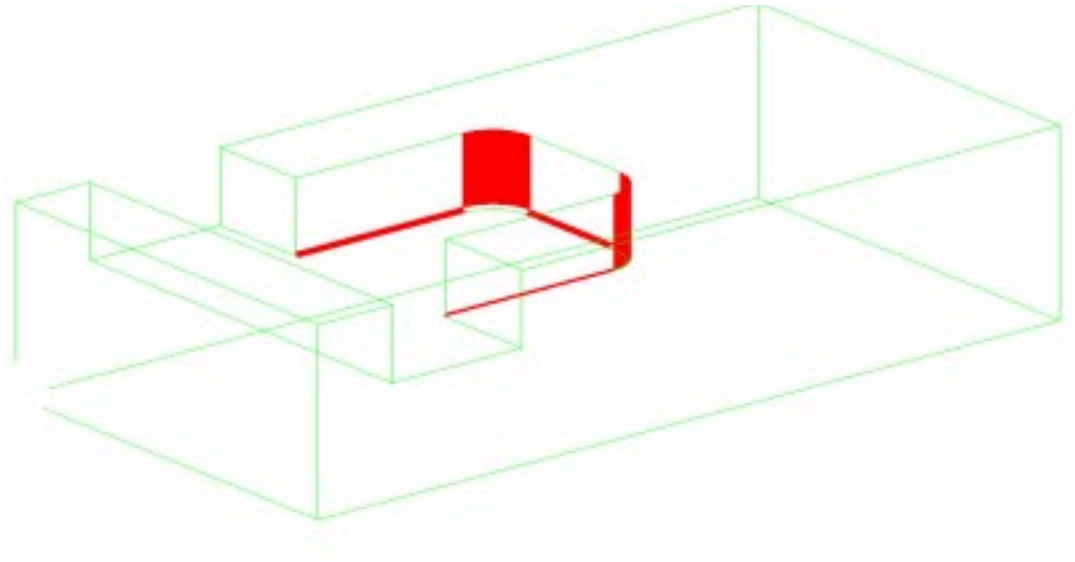


Figura 5.21: Faces cilíndricas de um rebaixo

Para verificar se o off-set das faces está contido no corpo é feita a operação de subtração entre o off-set da face e o volume do corpo.

A Figura 5.22 apresenta o off-set de todas as faces; se qualquer uma das faces geradas pelo off-set não estiver totalmente contida dentro do corpo, este método retornará um valor falso indicando um *pocket* inválido.

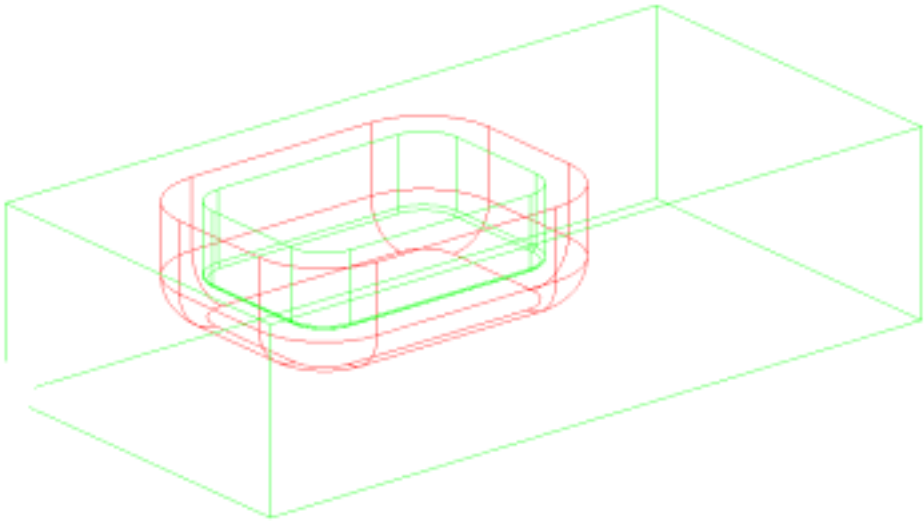


Figura 5.22: Verificação da validação de um pocket pelo off-set.

5.5.3 Método para dar nome às faces

```
F_Pocket::set_face_names();
```

Para cada face da *feature pocket* é dado um nome como atributo de face. Esta informação é utilizada na verificação da validação da *feature* e também para a transferência da entidade face ao arquivo STEP. Ainda neste método, é feita a verificação de existência de uma face virtual (além da face de topo) que denota um *pocket* aberto, conforme discutido em [4.3].

O critério para dar nomes às faces leva em conta o sistema de coordenadas locais do *pocket*. A face localizada no eixo *x* positivo recebe o nome de FACE_1 e a face localizada no eixo *y* positivo o nome de FACE_2. As faces opostas recebem o nome de FACE_3 e FACE_4 respectivamente. A face de fundo do *pocket* recebe o nome de FACE_BOTTOM.

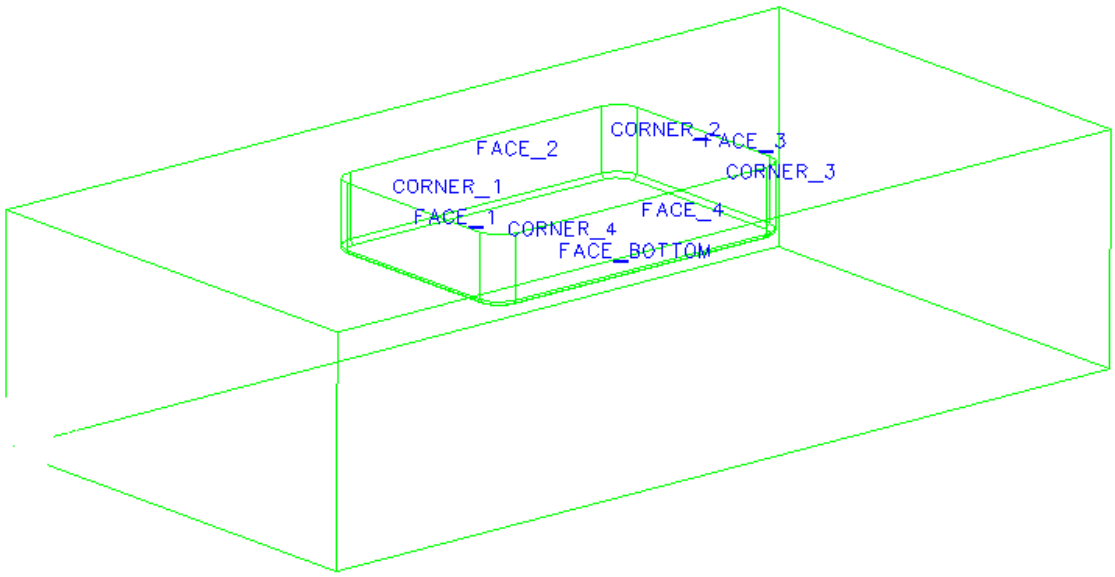


Figura 5.23: Nome das faces do Pocket

Também são dados nomes aos cantos do *pocket,* seguindo o mesmo critério. O canto localizado entre as faces FACE_1 e FACE_2 recebe o nome de CORNER_1, e assim sucessivamente, como pode ser verificado na Figura 5.23.

O diagrama de estrutura geral para este método é apresentado na Figura 5.24.

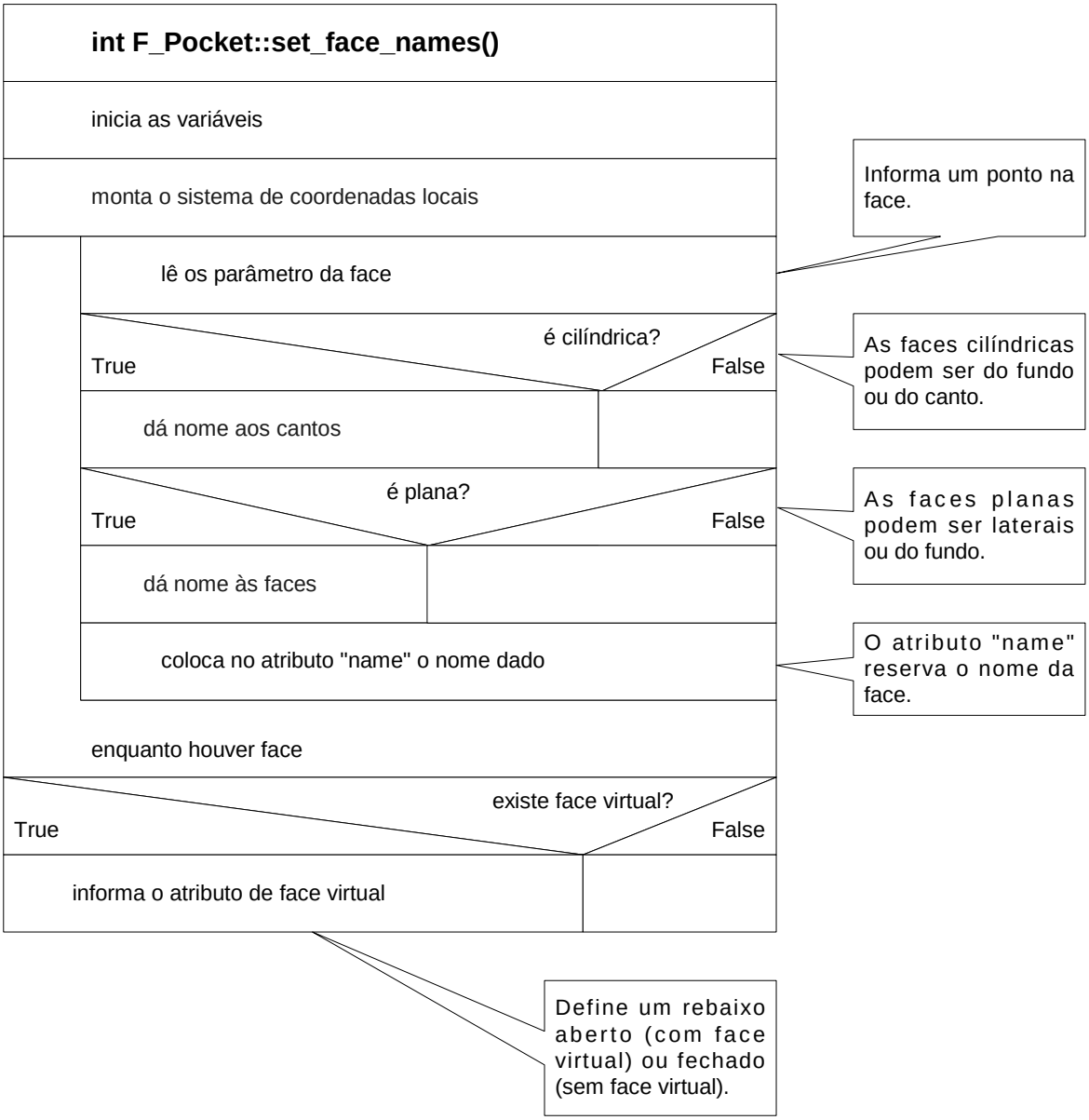


Figura 5.24: Diagrama de estruturas para o método set_face_names

O diagrama de escolha do nome das faces é apresentado na Figura 5.25.

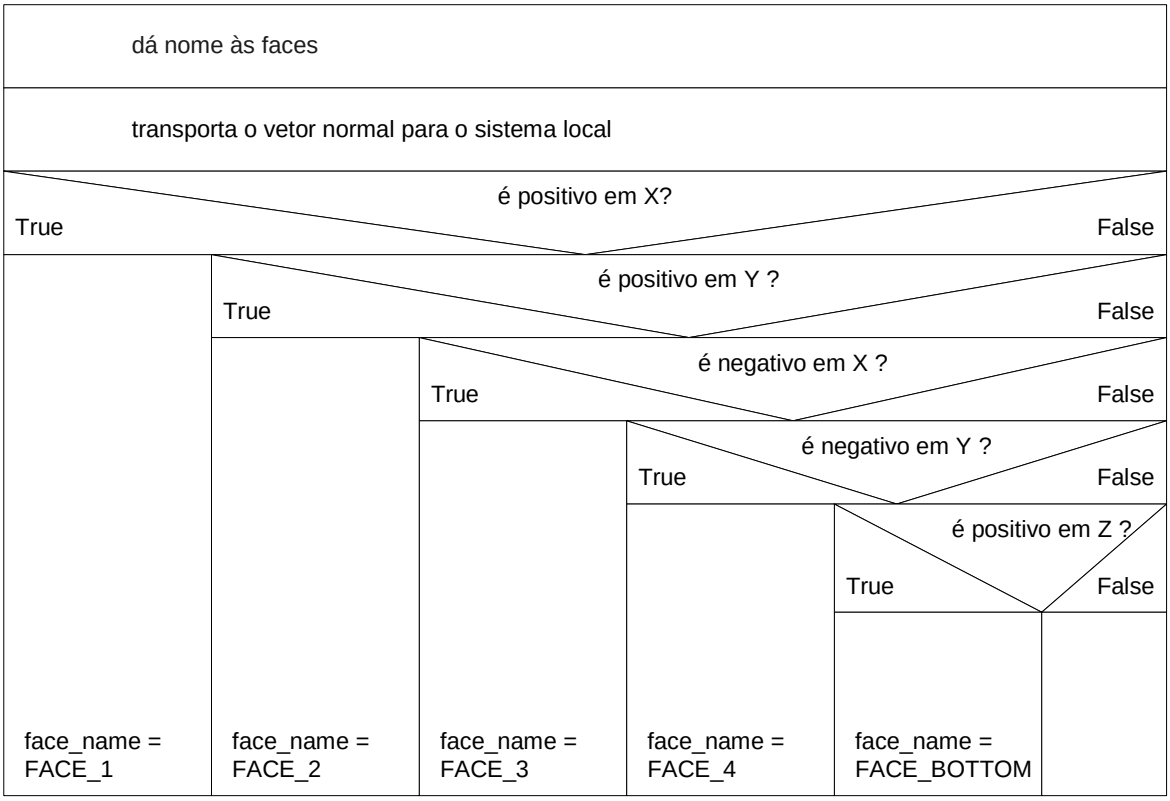


Figura 5.25: Diagrama para escolha do nome das faces.

O laço de repetição para as faces da *feature segue* enquanto houver faces, ou seja, enquanto o ponteiro para a lista de faces não retorne um valor nulo. São consideradas faces, as superfícies planas, enquanto que as superfícies cilíndricas são nomeadas como cantos. As superfícies esféricas, que fazem as intersecções entre os cantos e o fundo do *pocket* não são nomeadas, bem como as faces cilíndricas que fazem a intersecção entre as faces laterais e o fundo.

5.5.4 Método para Edição da *FEATURE POCKET*

F_Pocket::edit()

Para a alteração de parâmetros ou mudança de localização de um *pocket,* foi criado um método para edição, que está descrito na superclasse F_Feature, como virtual e sua implementação é feita para cada *feature*. Por não estar disponibilizada a função que faz a edição através de uma API do Unigraphics, a implementação deste método faz uma pseudo-edição da *feature*.

A edição fornecida por este método consiste na supressão de um *pocket* existente e na criação de um novo, com novos parâmetros, conforme pode ser visto no diagrama de estrutura da Figura 5.26.

F_Pocket::edit()
Inicializa as variáveis
Lê os valores do pocket existente
Apresenta o menu para novos dados
Lê parâmetros complementares
Apaga o pocket anterior
Cria um novo pocket

Figura 5.26: Diagrama de estruturas da edição de um rebaixo.

Os parâmetros essenciais para a edição de um *pocket*, lidos durante a execução deste método são: largura, comprimento, profundidade, raio de canto, raio de fundo. Os parâmetros complementares indicam a face de criação, o sentido e a referência de direção. Outras variáveis armazenam o posicionamento do *pocket*.

Para o tratamento destas variáveis, há a necessidade de conversão dos valores em ponteiros para caractere para valores reais, visto que o Unigraphics retorna os parâmetros como ponteiros para caractere e na criação são utilizados valores do tipo real. Outra peculiaridade deste método é que a função que lê os parâmetros de uma *feature* retorna as dimensões com uma cadeia de caracteres na forma: “p3 = 15.6”, sendo preciso isolar o valor desta dimensão para o correto tratamento da variável.

Com os parâmetros obtidos, é disponibilizada ao usuário uma caixa de diálogo para alteração destes valores e com estes é feita a criação de um novo *pocket* com a supressão do anterior.

5.5.5 Método de transferência para o arquivo STEP

```
CTransferObject* F_Pocket::get_step_param()
```

Este método retorna todos os parâmetros necessários para a construção do arquivo STEP. As informações do rebaixo são armazenadas em um objeto do tipo "CTransferObject" que pode ser entendido com uma caixa receptora de várias informações, e as transporta para o banco de dados estendido que dará origem ao arquivo físico STEP.

O diagrama de estrutura deste método é mostrado na Figura 5.27.

CTransferObject* F_Pocket::**get_step_param**
Inicialização das variáveis
Chamada dos parâmetros comuns
Definição das variáveis
Leitura das variáveis
Transferência para o objeto de transferência dos parâmetros (CTransferObjet)
Finalização

Figura 5.27: Diagrama de estrutura de inicialização de variáveis.

- Inicialização das variáveis

Duas variáveis são inicializadas para o método de transferência de dados para o arquivo STEP: a variável **retval,** que armazena a condição de retorno para todas as funções do UG (0 se não houver nenhum erro) e a variável **param,** um ponteiro para o objeto que serve de transporte para as informações.

- Chamada dos parâmetros comuns

As informações que são comuns a todas as *features* como: direção, localização e faces são inseridas no objeto de transporte através do método “get_step_param”, definido na superclasse abstrata para todas as *features*: “F_Feature”. Ao se executar esta linha de comando, os parâmetros comuns são lidos pelo método existente na superclasse e inseridos no objeto de transporte.

- Leitura dos parâmetros específicos

As informações específicas do *pocket* como altura, largura, comprimento, raio de canto e raio de fundo são lidas através de funções do UG.

As outras informações, como, por exemplo, sobre as tolerâncias, ficam armazenadas no banco de dados estendido e são recuperadas através da leitura de atributos vinculados à *feature*.

- Transferência das Informações

Todos os valores lidos são, então, transferidos para o objeto de transferência que é a variável de retorno deste método.

Cabe ressaltar que os parâmetros lidos através da função do UG são retornados com o tipo ponteiro para caractere e precisam ser convertidos para o tipo “double”, com o uso do método F_TOOLS_str2dbl.

Os atributos transferidos para o arquivo STEP dependentes, exclusivamente da *feature pocket,* estão definidos na Figura 5.28.

```
pocket->closed_boundary_(profile)
    profile->profile_length_(StepParam)
    profile->profile_width_(StepParam)
    profile->corner_radius_(StepParam)
pocket->bottom_condition_(bottom)
    bottom->floor_radius_(StepParam)
    bottom->floor_location(StepLocation)
    bottom->floor_normal(StepDirection)
pocket->pocket_depth_(lin_path)
    lin_path->distance(StepParam)
    lin_path->placement()
pocket->change_in_boundary(taper)
    taper->angle_(StepParam)
pocket->placement_(orientation)
```

Figura 5.28: Atributos da feature Pocket transferidos para o arquivo STEP

5.5.6 Método para Obtenção das *Features* que Interagem por Face

```
uf_list_p_t F_Pocket::get_faceint_features()
```

Este método retorna uma lista com as *features* que interagem por face com o *pocket*.

Para a busca da *feature,* que faz a interação por face com o *pocket,* são obedecidos os seguintes passos:

- Localização das faces laterais;
- Obtenção das duas arestas verticais de cada face lateral;
- Obtenção, dentre as arestas verticais, as mais próximas do centro canônico do *pocket*.

A *feature* que compartilha com o *pocket* destas arestas é a *feature* desejada, conforme discutido em [4.4.2].

Na Figura 5.29, é apresentado um exemplo em que as arestas verticais das faces paralelas estão representadas pelos números 1,2,3 e 4, e as arestas 2 e 4 são as arestas mais próximas do centro do *pocket*. Pelo fato destas arestas compartilharem com o *pocket* a *feature* rasgo, esta última é tida como *feature* que interage por face com o *pocket*.

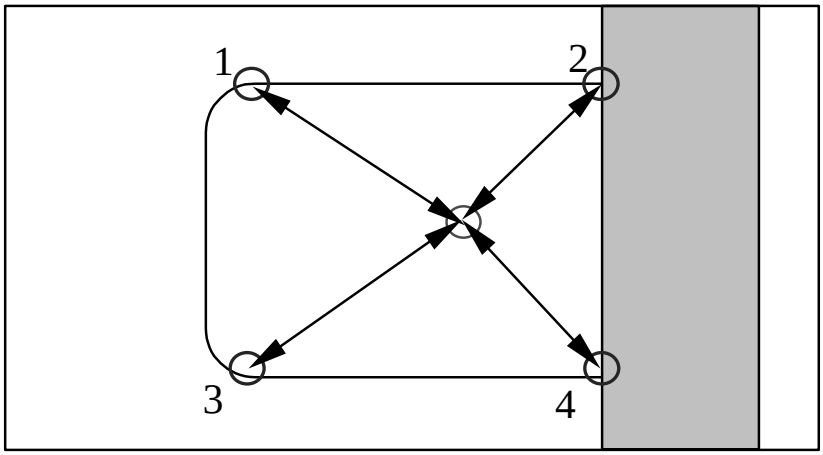


Figura 5.29: Verificação das arestas em relação ao centro do pocket.

A execução deste método pode ser dividida em três partes, a primeira em que é feita a identificação das faces laterais, a segunda, na qual são obtidas as arestas verticais mais próximas do centro do *pocket*, e a última que obtém as *features* que compartilham esta face.

Na primeira parte do método, quando é feita a identificação das faces laterais, é lançada mão do atributo “nome da face”. Se for notada a falta da face “FACE1” ou da face “FACE3”, as faces laterais são: “FACE2” e “FACE4”. Ao contrário, se a face que falta é a face “FACE2” ou “FACE4”, as faces “FACE1” e “FACE3” são as faces laterais.

Determinadas as faces laterais do *pocket* aberto é chamado o método “get_vertical_edges” para a determinação das arestas verticais destas faces, e então são calculadas as distâncias destas arestas ao centro do *pocket*. As mais próximas são colocadas em uma lista de arestas e enviadas para o método que retorna as *features* cujas arestas pertencem (“get_edge_*features*”).

O retorno desta função é uma lista com as *features* que compartilham arestas com o *pocket* e, portanto interagem por face com o *pocket* de modo direto.

5.5.7 Método para obtenção das arestas verticais

```
uf_list_p_t F_Pocket::get_vertical_edges(uf_list_p_t
face_edges)
```

A informação das arestas verticais é auxiliar na determinação das *features* que interagem por face com um *pocket* aberto, a *feature* que compartilha estas arestas tem uma interação por face com o *pocket*.

Este método obtém uma lista com as arestas verticais, tomando como plano horizontal a face de fundo do *pocket*. Este método é chamado pelo método “F_Pocket::get_faceint” somente se for verificada a condição de pocket aberto para verificação da *feature* que interage por face.

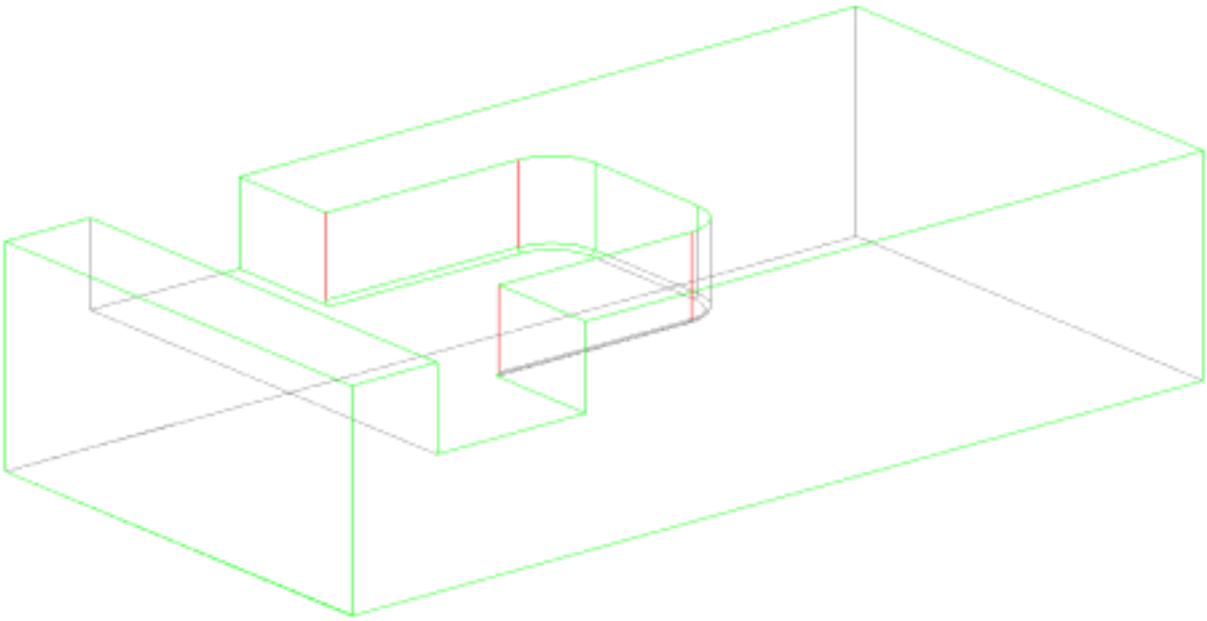


Figura 5.30: Arestas verticais de um pocket aberto.

Na Figura 5.30, são apresentadas, em vermelho, as arestas verticais das faces laterais do *pocket*.

O diagrama de estrutura deste método é apresentado na Figura 5.31.

A verificação de cada aresta é feita através da comparação com o vetor normal do *pocket*, portanto antes de se iniciar a verificação é preciso atualizar este vetor.

A varredura pelas arestas é feita por um laço que percorre a lista de arestas. De cada aresta são lidos seus vértices e a subtração dos vértices dá o vetor de cada aresta que é então comparado ao vetor normal. Se os vetores forem paralelos à aresta, é inserida na lista de arestas que será retornada pelo método.

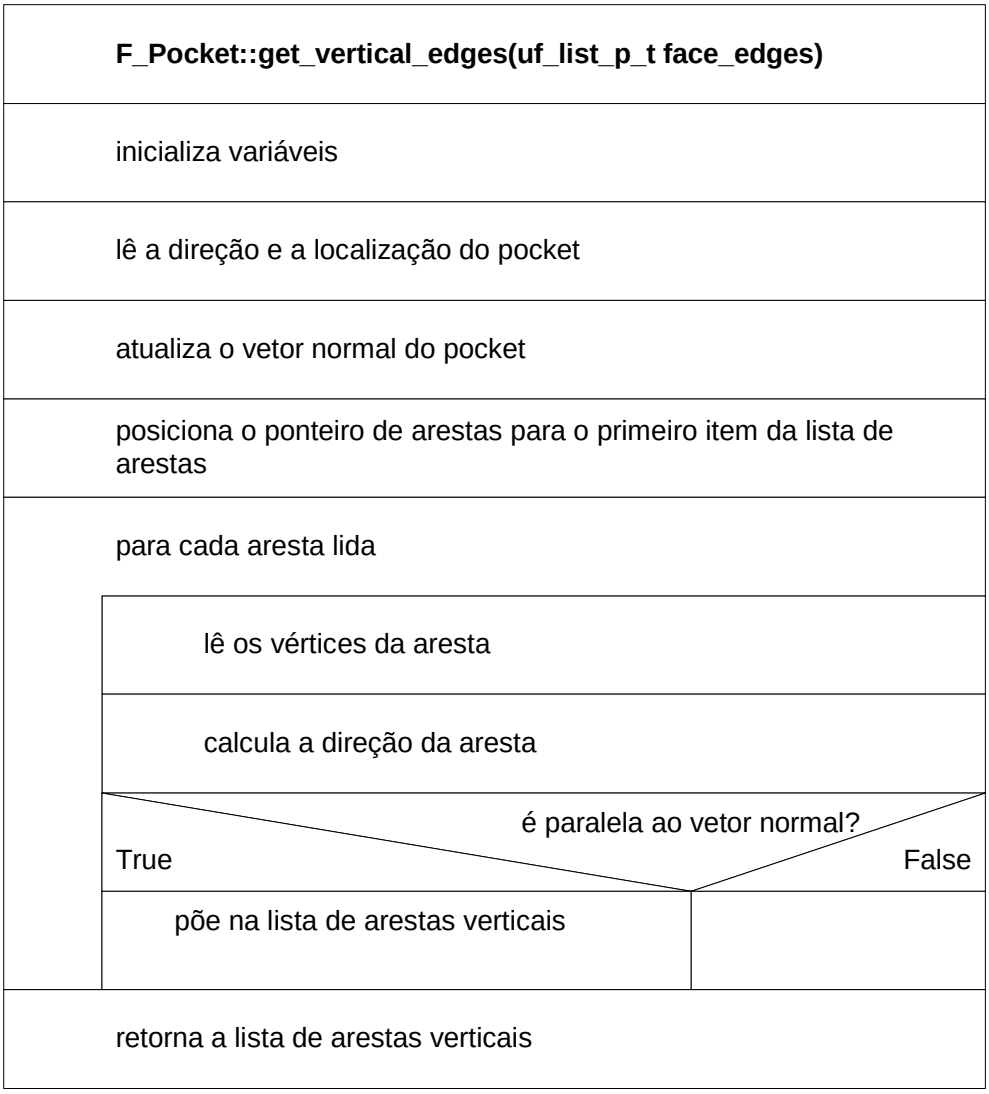


Figura 5.31: Diagrama de estrutura para obtenção das arestas verticais

5.6 Implementação das bibliotecas para utilização das normas STEP

Uma vez que o modelo do produto contemple o conjunto de informações necessárias para que haja a integração digital durante o seu desenvolvimento é preciso fazer com que estas informações sejam armazenadas em um formato neutro, de modo a garantir que qualquer outro sistema que utilize como base as normas STEP.

Para atingir este objetivo foram utilizados dois projetos de software: a Interface Padronizada de Acesso a Dados - SDAI e a biblioteca de classes STEP - SCL.

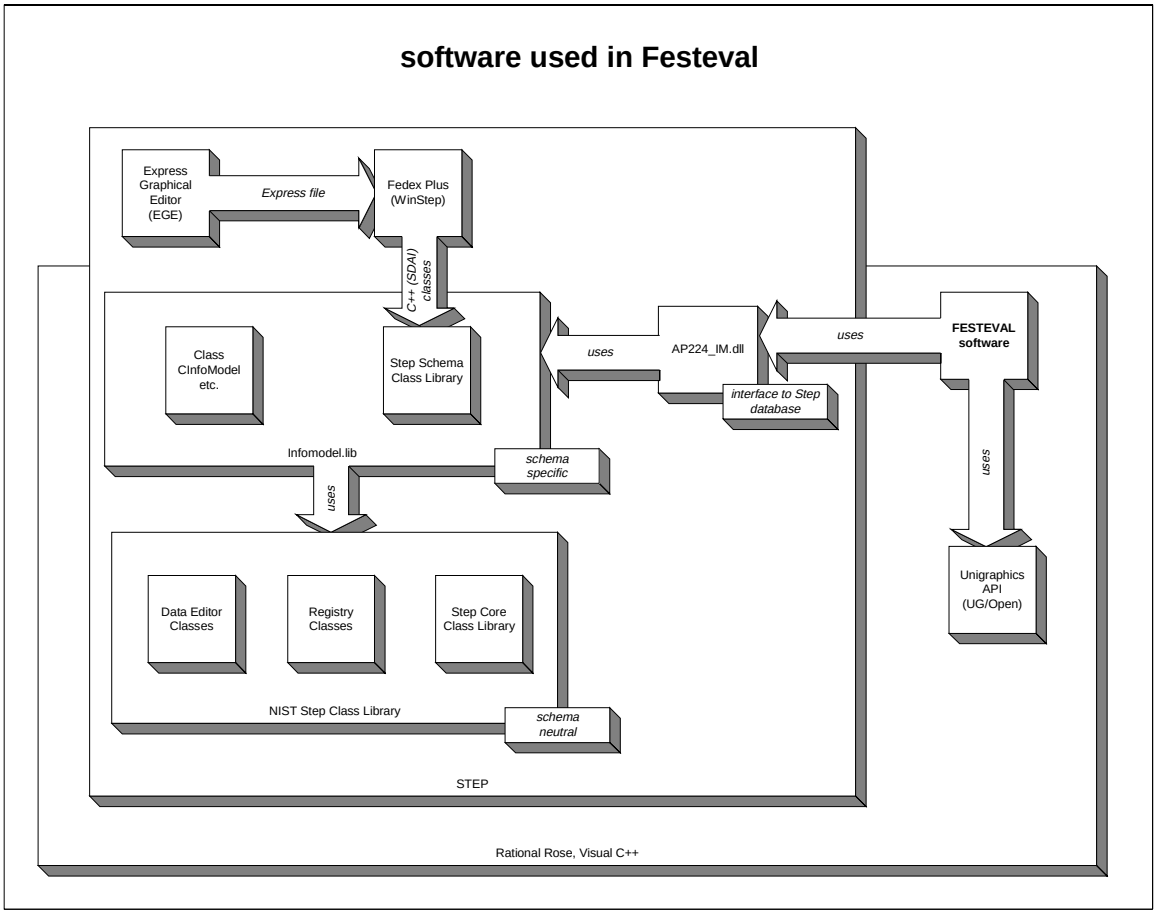


Figura 5.32: Esquema geral dos softwares utilizados no projeto.

5.6.1 SDAI - "Standard Data Access Interface"

A interface padronizada de acesso a dados é definida na seção 22 das normas STEP e definem os métodos para a transferência de dados. Este métodos são implementados em um linguagem de programação específica como C, C++, Java e CORBA/IDL (*Common Object Request Broker Architecture/Interface Definition Language*) [67]. Neste trabalho será apresentada a implementação de uma API - Interface de Aplicação de Programação ("Application Programming Interface") - para os dados definidos em EXPRESS com C++, definida na seção 23 das normas STEP.

As interfaces SDAI podem ser geradas de duas maneiras: básica ou avançada. Na implementação básica, as funções de acesso da API são definidas já no modelo de informação em EXPRESS. As interfaces criadas no modo básico são

implementadas por um compilador de EXPRESS que lê o modelo e gera o código que implementa funções para acesso aos dados. Na criação da interface de modo avançado, a API fica sempre independente dos modelos de informação usados para representação dos dados [68],[65].

O código gerado deve permitir a criação de instâncias das classes, inserir dados e registrá-los em um arquivo físico STEP, permitindo ainda a leitura de um arquivo físico e apresentando o conteúdo deste como instâncias das classes [65].

5.6.2 SCL - “STEP Class Library”

A SCL (STEP Class Library) é um conjunto de bibliotecas de classes em C++ criadas pelo “NIST - National Institute of Standards and Technology” que permite a representação da informação de acordo com a especificação de dados do EXPRESS (ISO 10303-11). As bibliotecas podem ser utilizadas para a construção de um programa executável em C++ nas aplicações que usam informações contidas em um arquivo EXPRESS [69]. As bibliotecas contêm as entidades como um dicionário do esquema de informações do EXPRESS e funções para a representação e manipulações de instâncias dos objetos descritos no arquivo EXPRESS. Aplicações simples como a gravação e recuperação de dados do EXPRESS em arquivos na forma descrita nas normas STEP - Parte 21, podem ser facilmente elaboradas com a SCL [41].

O desenvolvimento da SCL teve como objetivo atingir uma série de propostas, a mais importante foi ser útil para a execução final dos conceitos das normas STEP, como a implementação dos métodos e a criação de software. A criação da SCL teve interações com as seguintes atividades: desenvolvimento das normas STEP, desenvolvimento da SCL baseada nas normas STEP e verificação das aplicações através de implementações concretas [49].

No projeto *FESTEVAL,* a utilização das ferramentas do NIST foi feita através do software WinSTEP, desenvolvido pelo *Laboratory for graphical data processing of University of the Federal Armed Forces - Munich*. Este software é composto por um conjunto de ferramentas para utilização da linguagem EXPRESS, que auxilia em: edição de arquivos EXPRESS, compilação de arquivos EXPRESS,

transformação de arquivos EXPRESS em código C++ e transformação de arquivos EXPRESS no formato HTML.

O WinSTEP fez com que as ferramentas fornecidas pelo NIST fossem integradas em um ambiente gráfico baseado nas classes fundamentais da Microsoft (MFC) de maneira que as ferramentas possam se comunicar em janelas comuns, facilitando o seu uso [25].

A base do desenvolvimento do software com uma aplicação STEP é a transformação do protocolo de aplicação das normas STEP em uma biblioteca de classes em linguagem de programação [49], possibilitando assim a criação de instâncias das entidades definidas no protocolo de aplicação com os seus atributos, bem como a transposição destas instâncias em um arquivo físico STEP [65]. Neste trabalho será discutida a forma de atingir o objetivo descrito acima utilizando as ferramentas criadas pelo NIST, bem como a aplicação prática executada no projeto *FESTEVAL*.

5.6.3 Criação das Classes em C++ a partir dos diagramas em EXPRESS-G

Para a criação das classes em C++, foi feita a conversão do arquivo em linguagem EXPRESS através do gerador de código “fedex_plus” do kit de ferramentas criado pelo NIST. Cada entidade existente no protocolo de aplicação gera uma classe em C++. Na Figura 5.33, pode ser vista uma parte do código em C++ que apresenta, na classe “hole”, os métodos “edge_radius” que dão acesso a um parâmetro numérico. Nota-se que o método é sobrecarregado pelo seu argumento podendo, ser utilizado para escrever um valor no atributo (argumento “Numeric_Parameter”) ou para ler o valor do atributo (sem argumento). Da mesma forma, na Figura 5.34 podem ser vistos os métodos para ler e escrever os valores dos atributos: profundidade, conicidade e condição de fundo [51].

```
class SdaiHole : public SdaiMachining_feature{
...
SdaiHole ();
SdaiHole (SCLP23(Application_instance) *se, int
*addAttrs=0);
int opcode() {return 205;}
const SdaiNumeric_parameter_ptr edge_radius_() const;
void edge_radius_ (const SdaiNumeric_parameter_ptr x);
...
```

Figura 5.33: Código em C++ com a classe Hole e o método edge_radius

```
class SdaiRound_hole : public SdaiHole {
...
SdaiRound_hole ( );
SdaiRound_hole (SCLP23(Application_instance) *se, int
*addAttrs =0);
SdaiRound_hole (SdaiRound_hole& e);
~SdaiRound_hole();
...
public:

int opcode() {return 343};
const SdaiCircular_closed_profil_ptr diameter_() const;
void diameter_ (const SdaiCircular_closed_profil_ptr x);

const SdaiLinear_path_ptr hole_depth_() const;
void hole_depth_ (const SdaiLinear_path_ptr x);

const SdaiTaper_select_ptr change_in_diameter_() const;
void change_in_diameter_ (const SdaiTaper_select_ptr x);
...
```

Figura 5.34. Código em C++ com a classe Round_Hole e alguns métodos

5.7 Transferência dos dados para o banco de dados estendido

A criação da instância de um objeto é feita em um banco de dados preparado para a geração do arquivo físico STEP e cada instância será representada por uma linha no arquivo físico. O método utilizado para esta criação é o “Create_Entity”, da biblioteca de classes AP224IM. Na Figura 5.35, pode-se ver, na primeira linha, a criação da instância da classe “SdaiRound_hole”, em cuja linha no arquivo físico será iniciada pelo texto: “ROUND_HOLE”; esta criação usa como parâmetros um objeto do modelo de informações (CinfoModel* IM) e um objeto de transferência (CtransferObject* param). Na segunda linha, é feita a criação da instância da classe “SdaiCircular_closed_profil”, utilizando o parâmetro do modelo de informações. Na terceira linha, é feita a leitura do diâmetro através da recuperação da informação inserida no objeto da classe “CTransferObject”, que fica armazenada em “StepParam”.

Nas linhas 4 e 5, são feitas as inserções do atributo diâmetro na instância de “SdaiCircular_closed_profil” e desta na instância de SdaiRoundHole [70].

```
STEPentity* Create_RoundHole (CInfoModel* IM, CTransferObject* param)
```

```
{
1 SdaiRound_hole* hole= (SdaiRound_hole*)IM->CreateEntity
("Round_Hole");
2 SdaiCircular_closed_profil* profil = (SdaiCircular_closed_profil*
IM->CreateEntity("Circular_Closed_Profil");
3 StepParam = NumericParameter(IM, param->
NumericParameter("Diameter",0);
4 profile ->diameter_(StepParam);
5 hole ->diameter_(profile);
```

Figura 5.35: Código em C++ para criação de uma instância Round_hole e parâmetros

5.8 Transferência do banco de dados estendido para o arquivo físico

Com todas as informações inseridas no banco de dados como atributos das instâncias das classes definidas no protocolo de aplicação é feita a transferência destas informações para um arquivo texto, estruturado de acordo com a descrição das normas STEP, conforme pode ser visto na Figura 5.36. A linha 127 tem o texto “ROUND_HOLE” e indica em seu sexto atributo a linha 128 que tem o texto “CIRCULAR_CLOSE_PROFIL” e indica a linha 129, que, por sua vez, tem o texto “NUMERIC_PARAMETER” e indica um diâmetro de 10 mm.

```
ISO-10303-21;
HEADER;
FILE_DESCRIPTION((‘Step Physical File’),’Level 1.0’);
FILE_NAME(C:\UGS160\UGII\furo’,’2001-02-22T11:14:59’,(‘Current User’);
FILE_SCHEMA((‘HSCWF’));
ENDSEC;
DATA;
#0=PART($,$,#2,$,$,$,$,$,$,#1,$);
#1=FEATURE_LIST((#127));
#2=SHAPE($,$,#3);
...
#127=ROUND_HOLE((#42,#101),$,#132,#148,#147,#128,#130,$,$);
#128=CIRCULAR_CLOSED_PROFIL($,#129);
#129=NUMERIC_PARAMETER(‘Diameter’,‘mm’,10.);
#130=LINEAR_PATH(#132,#131);
...
ENDSEC;
END-ISO-10303-21;
```

Figura 5.36: Arquivo físico STEP com as informações do furo

6 Conclusões

Dentro dos estudos efetuados no projeto de pesquisa, foi observada uma constante busca, na maior integração nas etapas do desenvolvimento do produto com o maior uso de ferramentas computacionais. Para este desenvolvimento, as melhorias nos sistemas computacionais e nos sistemas de desenvolvimento de software têm sido intensamente aproveitadas.

Nesta dissertação, foram utilizadas, conjuntamente, duas tecnologias principais: a tecnologia *features,* para a criação do modelo do produto e a tecnologia das normas STEP, para armazenar os dados deste modelo. Como apoio a estas tecnologias, foi utilizado o desenvolvimento de software orientado a objetos modelado com UML.

A utilização conjunta destas quatro tecnologias mostrou um aproveitamento excelente. Principalmente porque o paradigma da orientação a objetos foi utilizado tanto para o modelo do produto, como para o modelo do software e para o arquivo dos dados do modelo segundo as normas STEP.

A analogia entre o conceito de *features* e o conceito de objetos é bastante significativa e foi, verdadeiramente, a base para o desenvolvimento deste projeto de pesquisa.

O início deste trabalho foi a especificação do modelo comum de dados, de modo que as *features* utilizadas na criação do modelo são derivadas de uma biblioteca de classes já existentes e descritas de acordo com as normas STEP e este foi o ponto inicial do desenvolvimento deste trabalho. A partir daí são criadas *features* com a mesma idéia de objetos que são transferidos para objetos do banco de dados o qual se tornará o arquivo físico de acordo com o modelo de dados que iniciou o processo.

A utilização de um padrão perene por todo o processo de criação do modelo do produto dá a garantia de um produto final compatível com as expectativas iniciais, pois permite a utilização completa do conceito de *features*

A utilização das normas STEP aproveita as vantagens de múltiplos centros de desenvolvimento. Para a produção do software utilizado neste projeto, foram

utilizados, simultaneamente, projetos de software desenvolvidos em quatro institutos de pesquisas de três países (Alemanha, EUA e Brasil). Isto só foi possível devido à padronização no desenvolvimento dos projetos e à documentação disponibilizada para sua utilização.

A participação ativa de centros de pesquisas e de um usuário final obrigou um desenvolvimento em parceria que efetivamente atendesse aos anseios de um usuário real, trazendo para bem próximo, a utilização real do modelador criado.

Como resultado deste trabalho tem-se a criação de um modelo do produto com um modelador melhorado de forma a atender os requisitos da tecnologia *features*; e a transcrição deste modelo em um arquivo físico STEP que permite a transferência de informações de modo a garantir a integração digital do desenvolvimento do produto.

6.1 Sugestões para trabalhos futuros

Para a continuidade deste trabalho é sugerida a implementação da construção do modelo no UG através da leitura de um arquivo STEP, fazendo o caminho inverso e verificando a eficiência da troca de dados. O protótipo do modelador criado ficará completo com a implementação de todas as *features* contempladas no modelo de informações bem como de todos os atributos tecnológicos.

7 Referências Bibliográficas

7.1 Bibliografia Citada

[1] HENRIQUES, J. R.; SCHÜTZER, K. Os desafios enfrentados para superar as limitações atuais na troca de dados entre sistemas CAD através da STEP. In: Anais do V Encontro de Mestrandos e I Encontro de Doutorandos em Engenharia, Universidade Metodista de Piracicaba, Piracicaba, 2001, pp. 101 -111.

[2] DEBONI, J. E. Z.; MARTINI, J. S. C. Um método para desenvolvimento de sistemas de apoio a projetos de engenharia. São Paulo: EPUSP, 1997. 15 p. Boletim Técnico da Escola Politécnica da USP BT/PCS/9703.

[3] PRESSMAN, R. Engenharia de Software. Tradução José Carlos B. dos Santos. Makron: São Paulo. 1998. p.1056

[4] FOWLER, M. UML Distilled / A Brief Guide to the Standardization, Atlanta: Addison-Wesley, 2000.

[5] WILSON, P. R. A short history of CAD data transfer standards. IEEE Computer Graph. & Aplic. v.6. n. 6 (1987) p. 64-7

[6] MANTYLA, M. An introduction to solid modeling. 401p. Marco Markovit: 1998

[7] BADIOU, A. Sobre o conceito de modelo. Tradução: Fernando Bello Pinheiro. São Paulo: Mandacaru, 1989. 130 p.

[8] TENÓRIO, R. M. Cérebros e computadores: A complementaridade analógico-digital na informática e na educação. São Paulo: Escrituras, 1998. 211 p. (Ensaios Transversais).

[9] SCHÜTZER, K.; HENRIQUES, J. R. Como as Normas STEP Podem Integrar a Modelagem do Produto. Revista Máquinas e Metais, São Paulo, v. 37, n.426, pp. 100-117, 2001.

[10] National Institute of Standars and Technology. Computer - Integrated Construction Group. O que é STEP. Disponível em: <http://cic.nist.gov/plantstep/stepinfo/step_def.htm> acesso em 20 nov. 2000.

[11] HOLLAND, W.; BRONSVOORT, W.F. Assembling *features* in modeling and planning. Robotics and Computer Integrated Manufacturing, n. 16, p. 277-294. Pergamon: 2000

[12] DAS, D.; GUPTA, S.K.; NAU, D.S. Generating redesign suggestions to reduce setup cost: a step towards automated redesign.Computer-aided Design, v.28, n.10, p. 763-782. Elsevier: 1996

[13] ACHTEN, H. H.; LEEUWEN, J. P. VAN. 1999, Feature-based high level design tools: a classification, in G. Augenbroe and C. Eastman (eds), Computers in Building: Proceedings of the CAADfutures'99 Conference, Kluwer, Dordrecht, pp. 275-290

[14] CHEN, Y.; WEI, C. Computer-aided feature-based design for net shape manufacturing. Computer integrated manufacturing systems. Elsevier, 1997. 10 v. p.147-164.

[15] FENG, C-X.(J.); KUSIAK, A.; HUANG, C-C. Cost evaluation in design with form-*features*. Computer-aided design, v.28, n.11, p. 879-885. Elsevier: 1996

[16] HOLLAND, M. Product data thecnology and STEP: Comprehensive standards are prerequisites for the effective development of products and effective business process. s.l., s. n, 1998.

[17] WACO, D. L., KIM, Y. S. Considerations in positive to negative conversion for machining *features* using convex decomposition, ASME Computers in Engineering Conference. In Proceedings…p. 35-46. 1993

[18] KAMRANI, A.K. An integrated knowledge based system for product design feasibility and manufacturability analysis. Computer Ind. Engineer, v. 31 n.1/2, p.83-86. Elsevier: 1996

[19] DONG, J.(J.); PARSAEI, H.R.; LEEP, H.R. Manufacturing process planning in a concurrent design and manufacturing environment. Computer Ind. Engineering, v. 3, n.1, p. 83-93. Elsevier: 1996.

[20] GLOCKNER, C. Basic software tool of the feature modeller for the demonstrator. PTW-Dvd-03. Technical Report INCO-DC Project 96.2161. 1998.

[21] VANCZA, J.; MARKUS, A. Experiments with the integration of reasoning optimization and generalization in process planning. Advances in Engineering Software n. 2, p. 29-39. Elsevier: 1996.

[22] BELAZIZ, M.; BOURAS, A.; BRUN, J.M. Morphological analysis for product design. Computer-aided Design, v.32, p. 377-388. Elsevier: 2000

[23] DÜRR, H.; SCHRAMM, M. Feature based feedback into the early stages of design. European Journal of Operational Research, n. 100, p. 338. 350. Elsevier: 1997.

[24] HOUNSELL, M.S.; CASE, K. Morphological and volumetrical feature-based designer's intent. 1997, 13th National Conference on Manufacturing Research - Advances in Manufacturing Technology XI - Glasgow, v.1, p. 65-8.

[25] SCHÜTZER, K. et al. Improved software tools of the feature modeller software tools for recognition of most geometrical and technological interdependencies. SCPM-Dvd-14AC. Technical Report INCO-DC Project #96-2161. 2000.

[26] REGLI, W.; PRATT, M.J. What are *features* interactions. The 1996 ASME Design Engineering Technical Conference and Computers in Engineering Conference. In Proceedings...

[27] NOORT, A.; BIDARRA, R.; BROONSVOORT, W.F. Satisfying interactions constraints, 2000.

[28] BIDARRA, R.; BROONSVOORT, W. F. History-independent boundary evaluation for feature modeling. Proceeding of DETC'99, 1999 ASME Engineering Technical Conferences, Sept 12-15, 1999, Las Vegas, Nevada.

[29] BIDARRA, R.; BROONSVOORT, W.F. On families of objects and their semantics. In. Proceedings of Geometric Modelling and Processing, 2000. 10-12 April, Honk-Kong, China, IEEE Computer Society, CA, USA. p.101-111

[30] DANKWORT, C.W. Present-day demands versus long term open CAD/CAM strategies. Process in Chain Automotive Industry. Paris 1997.

[31] SCHULZ, H.; SCHÜTZER, K. Integração de Projeto e Planejamento Baseado em Feature. Máquinas e Metais. 27 (1993) 332, pp. 28-37.

[32] BROONSVOORT, W. et al. Essential developments in feature modeling. CAD/Graphics'2001. International Academic Publishers

[33] BURKETT, W.C. Product data markup language: a new paradigm for product data exchange and integration. Computer-aided design, v. 33, n.1, p. 489-500. Elsevier, 2000.

[34] Huang, G. Q. Design for X; Concurrent engineering imperatives , Chapman & Hall: 1996.

[35] ZHANG, S. G.; AJMAL, A.; WOOTON, J.; CHISHOLM, A. A feature-based inspection process planning system for coordinate measuring machine (CMM). Journal of Materials Processing Technology, n. 107, p. 111-118. Elsevier, 2000.

[36] NORRIE, D.H., FAUVEL, O.R. GAINES, B.R. "Object-Oriented Management Planning Systems for Advanced Manufacturing", International Journal of Computer Integrated Manufacturing, Vol. 3, No. 6, 1990, pp. 373-378.

[37] EVERSHEIM, W.; MARCZINSKI; G.; CREMER, R. "Structured Modelling of Manufacturing Processes as NC-Data Preparation", CIRP Annals 1991: Manufacturing Technology, Volume 40, Number 1, pp. 429-432

[38] SCHÜTZER, K. et al. Requirements of *features* specification of the feature-based library. SCPM-Dvd-02. Technical Report INCO-DC Project 96.2161. 1998.

[39] CONCHERI, G.; MILANESI, V. MIRAGGIO: a system for the dynamic management of product data and design models. Advances in engineering software n. 32. p. 527-543. Elsevier, 2001.

[40] HOFFMAN, C.M.; JOAN-ARINYO, R. CAD and the product master model. Computer-Aided Design, v. 30, p. 905-918. Elsevier: 1998

[41] SCRA: STEP application handbook - SCRA - International Boulevard North Charleston - SC - 2000

[42] USHER, J.M. A STEP based object oriented product model for process planning. Computer Ind. Engineer, v. 31 n.1/2, p.185-188. Elsevier: 1996

[43] MITCHELL, M.J. Capabilities for product data exchange. Disponível em: http://www.mel.nist.gov/msidlibrary/doc/mitchell96/stepalb2.html Acesso em: 21 fev. 2001.

[44] SHIN, Y.; HAN, S-H. Integration of heterogeneous CAD database using STEP and Internet. International Conference on Electronic Commerce - Seoul, 1997.

[45] BETTIG, B.; SHAH, J.J. An object-oriented program shell for integration CAD software tools. Advances in engineering software, n. 30. p.529-541. Elsevier, 1999.

[46] ALBERT, M. STEP-NC: The end of G-Codes? 2000.

[47] MA, S.; MARECHAL, Y; COULOMB, J.L. Methodology for na implementation of the STEP standard: a JAVA prototype. Advances in Engineering Software, v. 32, p. 15-19. Elsevier: 2000

[48] NELL, J.; NIST. STEP on page. Disponível em: <http://www.mel.nist.gov/sc5/soap/> acesso em 20 nov. 2000.

[49] LOFFREDO, D. Fundamentals of STEP Implementation. Disponível em <http://www.steptools.com/library/fundimpl.pdf> acesso em 27 nov 2000.

[50] KRAMER, T.R.; Proctor, F.M. Feature-based control machining center. NIST, 1996

[51] LEIBRECHT, S. Development of a design environment with STEP processor for a feature base 3D-CAD System. Universidade Metodista de Piracicaba. Diploma Thesis - 2000.

[52] ARNOLD, F. MACAO - A journey into CAx Interoperability and collaborative design, International Conference on Information Visualisation,, 2000

[53] SKOGAN, D. Rules for how to use EXPRESS. Disponível em: http://comelec.afnor.fr/servlet/ServletForum?form_name=cForumFich&file_name=Z13C%2FPUBLIC%2FDOC%2F287n512.doc&login=invite&password=invite

[54] KERN, V.M., BØHN, J.H. STEP databases for product data exchange. International Congress of Industrial Engineering. In. Anais v. III, p. 1337-1341. São Carlos, 1997.

[55] PEDROSA, B. M. Programação Orientada a Objetos com C++, Apostila para Curso de Férias, Unimep, 1998

[56] VERKEYEN, A.A.A. Disponível em: http://allserv.rug.ac.be/~averkeyn/documents/histlang.html. Acesso em: 15/05/2002.

[57] PRABHU, B.S.; BISWAS, S.; PANDE, S.S. Intelligent system for extraction of product data from CAD models. Computers in industry, n. 44 p. 79-95. Elsevier, 2001.

[58] ARNOLD, F.; PODEHL, G. Best of both worlds - a mapping from Express-G to UML Research group for computer application in engineering design. University of Kaiserslautern, Germany. 1999

[59] KHOSHAFIAN, S. Banco de Dados Orientado a Objetos, Rio de Janeiro: Infobook, 1994.

[60] CASE, K.; HOUNSELL, M.S. Feature modelling: a validation methodology and its evaluation. Journal of Material Processing Technology 4655, p. 1-9, Elsevier: 2000.

[61] CLAASSEN, E. Specification of the common data model with basic compatibilities. DiK-Dvd-04. Technical Report INCO-DC Project #96-2161. 1998.

[62] BETTIG, B.; SHAH, J.J. Derivation of a standard set of geometric constraints for parametric modeling and data exchange. Computer-aided design, n. 33, p. 17-33, Elsevier, 2001.

[63] LOPASSO, P.S.; TOLEDO; R.B.R.; BELLO, V.D. Validation and tests of the first and second prototype FestPlan Module. Technical Report INCO-DC Project #96-2161. 2000.

[64] AGOSTINHO, O. L. Tolerâncias, ajustes, desvios e análise de dimensões. São Paulo, Edgard Blücher. 1977. 295 p.

[65] SAUDER, D. A. MORRIS, K.C. Design of a C++ Softwares Library for implementing EXPRESS. The NIST STEP Class Library - Express User Group - 1995

[66] CLAASSEN, E. Specification of the representation of interdependencies in the common data model. DiK-Dvd-09. Technical Report INCO-DC Project 96.2161. 1999.

[67] SDAI C Reference - STEP Tools, Inc. Rensselaer Technology Park, Troy, New York 12180 - 1997.

[68] FRISCH, H. P. A Product data life cycle information management system. Infrastructure with CAD/CAE/CAM, Task automation, an intelligent support capabilities - NASA/Goddard Space Flight Center, Greenbelt, MD 20771, USA - 1998

[69] HARDWICK, M. STEP data exchange moves into implementation phase. Disponível em: http://www.steptools.com/library/stepimpl.html. Acesso em: <20 fev. 2001>.

[70] SCHÜTZER, K.; GARDINI, N.; FOLCO, J. Basic software tool of the feature modeller for the demonstrator. SCPM-Dvd-05. Technical Report INCO-DC Project #96-2161. 1998.

7.2 Bibliografia Consultada

ALBERT, M. Feature recognition - the missing link to automated CAM.

ATKINSON, M. et al. The object-oriented database system manifesto. In. Deductive and OODb Proceedings “DOOD’ 89 Kioto

AU, C.K.; YUEN, M.M.F. A semantic feature language for sculptured object modelling. Computer-aided design, n.32, p. 67-74. Elsevier: 2000

AVGOUSTINOV, N.; BLEY, H. Computer aided information for manufacturing software systems.

BAKER, R.P.; MAROUPOULOS, P.G. An architecture for the vertical integration of tooling considerations from design to process planning. Robotics and Computer Integrated Manufacturing, n. 16 p.121-131. Elsevier 2000.

BROUËR, N.; WECK, M. Feature-oriented programming interface for an autonomous production cell. Control Engineering Practice, n. 6, v. 6, p. 1405-1410. Pergamon, 1988

BRUNETTI, G.; GOLOB, B. A feature-based approach towards na integrated product data model including conceptual design information. Computer-aided design, n. 32. p.877-887. Elsevier: 2000

CANCIGLIERI JUNIOR, O.; COELHO, L.S.; YOUNG, R.I.M. Uma metodologia orientada a objeto no suporte a projetos orientados para a manufatura. Congresso brasileiro de engenharia de fabricação, 1, Curitiba. 2001.

CHEP, A. et al. Design of OO database for the definition of machining operation sequences of 3D workpieces. Computer Ind. Engng. V. 34, n. 2 p. 257-279

CHOLVY, L; FOISSEAU, J. ROSALIE: A CAD OO on rule-based system. Information Processing 83, Elsevier Science, 1983 p. 501-5

COSTA, H.A.X. BDOO: uma visão geral. Infocomp - Revista de Computação da UFLA. Semana de Ciência da Computação, 3. Universidade Federal de Lavras. Lavras nov. 2000. In Anais...

DAIMLERCHRYSLER User-oriented inspection for powertrain engineering. Notas de curso.

DANKWORT, C.W. Process chain in automotive industry: present-day demands versus long term open CAD/CAM strategies. 1997

FALEIRO JR., J. M. Um padrão comparativo para metodologias de desenvolvimento orientadas a objeto.

GLOCKNER, C.; SCHÜTZER, K. Specification of the required platform and evaluation of investigated tools. PTW-Dvd-01. Technical Report INCO-DC Project 96.2161. 1998.

GRABOWSKI, H. The development of Advanced Modeling Techniques CIM-Europe 1988 4th CIM-Europe Conference

GROSS, M. et al. Constraints: Knowledge Representation in Design. Design Studies, v. 9 n. 3 1980.

KILB, T.; ARNOLD, F. Data management in distributed CAx Systems. Research group for computer application in engineering design. University of Kaiserslautern, Germany. 1999

KLOCKE, F. et al. Methods and tools supporting modular process plan. Robotics and computer integrated manufacturing. Elsevier, 2000. 16 v. p.411-423.

KOJIMA, T. et al. An expert system of machining planning in Internet environment. Journal of materials processing technology, n. 107, p. 160-166. Elsevier, 2000.

LIANG, W-Y.; O'GRADY, P. Design with objects an approach to object-oriented design. Computer-aided design, v. 30, n. 12, p. 943-956. Elsevier, 1998

MALHOTRA, M. K.; HEINE, M.L.; GROVER, V. An evaluation of the relationship between management practices and computer aided design technology. Journal of operations management. v.19, p. 307-333, Elsevier, 2001

MANTIPRAGADA, R.; KINZEL, G.; ALTAN, T. A computer aided engineering system for feature-based design for box-type sheet metal parts. Journal of Material Processing Technology, v. 57, p. 241-248. Elsevier, 1996

MARTTII, M. An introduction to solid modeling. Maryland: Computer Science, 1988. 401 p.

NASSU, E. A. Banco de dados orientados a objetos. Tradução: Waldemar W. Setzer. Edgard Blucher: São Paulo. 1999. p 122

National Research Council - IMTI: Introducing STEP: the foundation for product data exchange in the aerospace and defense sectors. Disponível em <http://strategis.ic.gc.ca/STEPguide> acesso em 30 jan. 2001

OH, Y. HAN, S.; SUH, H. Mapping product structures between CAD and PDM systems using UML. Computer-aided design 2000.

PDES, Inc. Contact. Disponível em: <http://pdesinc.aticorp.org/> acesso em 27 nov. 2000.

PEDLEY, A.G. The potential to exchange feature models with user defined feature libraries. Journal of Materials Processing Technology, n. 61, p. 78-84. Elsevier: 1996

PEREIRA, G.M. Implementing the control of Product/Process Revisions in a Concurrent Engineering Environment.

PHILLIPS, L. Integrated construction group. ISO standards development in TC 184/SC4. Disponível em: < http://www.mel.nist.gov/msid/sima/stc/1c__lisa/sld001.htm > acesso em 20 nov. 2000.

ProSTEP Association. 4Th ProSTEP Benchmark. Disponível em <http://www.prostep.de/english/e_ev_rahens.html> acesso em 17 nov. 2000.

ProSTEP Association. 6Th ProSTEP Benchmark. Disponível em <http://www.prostep.de/e_psbenchmark_6.htm> acesso em 20 jan. 2001.

ProSTEP Association. Report on the first benchmark of STEP processors. Darmstadt, 6 maio 1996.

RUEGG, A. The importance of interfaces between CAD, CAM & CNC for technology. Annals of CIRP, v. 37-1 Paris, 1988

SALOMONS, O. Computer support in the design of mechanical products. 1995. 256 p. Thesis to get the PhD degree at the Universiteit Twente, Groningen. Disponível em: < www.opm.wb.utwente.nl/staff/otto/thesis > Acesso em: 02/04/2001.

SÁNCHEZ, J.M.; PRIEST, J.W.; SOTO, R. Intelligent reasoning assistant for incorporating manufacturability issues into the design process. Expert System with Applications, v.12, n. 1, p. 81-88. Elsevier, 1997

SANDIFORD, D.; HINDUJA, S. Construction of feature volumes using intersection of adjacent surfaces. Computer-aided design v. 33, p. 455-473. Elsevier, 2001.

SHAH, J.J.; DEDHIA, H.; PHERWANI, V.; SOLKAN, S. Dynamic interfacing of applications to geometric modeling services via modeler neutral protocol. Computer-aided design, v 29, n. 12, p. 811-824. Elsevier, 2000.

SRINIVASAN, V.; FISCHER, G.W. Direct interface integration of CAD and CAM Software - A milling application. Journal of materials processing technology, n. 61, p. 93-98. Elsevier 1996.

TANAKA, C. C. Metodologia orientada a objetos e sua aplicação em sistemas CAD baseado em features. Dissertação de Mestrado. Escola Politécnica - USP 1997

TAVARES, M.; BUCHER, L.H. Overview on product data exchange in supply chain management in agrobusiness. Juiz de Fora, 2000. Agrosoft - Softex. Disponível em: www.agrosoft.com/ag97/papers/w3w1030.htm. Acesso em: 15 ago. 2001.

TÖNSHOFF, H.K.; WOELK, O. Feature-based data model for integration design and process plant. ProSTEP Science Days 2000. Sept. 200. In... Proceedings. (Transparências da exposição)

WILSON, P. R. Information Modeling IEEE Computer Graph. & Aplic. v.7 n. 11 (1987) p. 58-61

YAN, X.; YAMAZAKI, K.; LIU, J. Recognition of machining feature and feature topologies from NC programs. Computer-aided design, v. 32 p. 605.616, Elsevier, 2000.

Anexo A - Programa Fonte

O programa fonte está disponível na versão digital desta dissertação, no CD anexo.

Anexo B - Pesquisas sobre modeladores baseados em *features*.

Logo	
Acrônimo	IMACS
Nome	Integrated Manufacturability Analysis and Critiquing System
Homepage	http://www.cs.umd.edu/projects/cim/imacs/imacs.html
Principal Pesquisador	Satyandra K. Gupta
Universidade	Institute for Systems Research, University of Maryland
Kernel	EDS/Unigraphics
Estilo	Reconhecimento
Arquivo	PDES/STEP
Ano	1996
Linguagem	C++

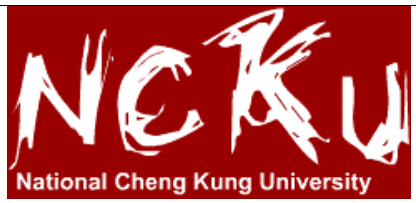
Logo	
Acrônimo	FRIEND
Nome	*Feature*-Based Validation Reasoning for Intent-Driven Engineering Design
Homepage	http://www.lboro.ac.uk/departments/en/research/cae/res_int/phds/m_hounsell.html http://www.joinville.udesc.br/departamentos/dcc/professores/marcelo/pesquisa.html
Principal Pesquisador	Marcelo da Silva Hounsell
Universidade	Loughborough University
Kernel	
Estilo	
Arquivo	
Ano	
Linguagem	

Logo	
Acrônimo	FROOM
Nome	*Features* and Relations used in OO Modelling
Homepage	http://www.opm.wb.utwente.nl/pt/projects/froom
Principal Pesquisador	Otto Salomons
Universidade	University of Twente

Logo	
Acrônimo	EXTDesign++
Nome	
Homepage	http://www.cs.hut.fi/~tla/publications/ASME96/ASME96TKO.html http://www.cs.hut.fi/~mam/extd.html
Principal Pesquisador	Timo Laakko
Universidade	Helsinki University of Technology

Logo	
Acrônimo	SPIFF
Nome	*Feature* Modelling System
Homepage	http://www.webspiff.org/
Principal Pesquisador	Rafael Bidarra
Universidade	Delft University of Technology

Logo	
Acrônimo	DEFEATOR
Nome	Design by *features* editor
Homepage	
Principal Pesquisador	Mark B. Unger
Universidade	National Institute of Standards and Tecnhology - NIST

Logo	
Acrônimo	
Nome	
Homepage	http://www.ncku.edu.tw/english
Principal Pesquisador	Yuh-Min Chen
Universidade	National Cheng Kung University

Logo	
Acrônimo	FREDS
Nome	*Features*-based Rapid Engineering Design System
Homepage	http://www.poly.rpi.edu/
Principal Pesquisador	M. J. Wosny
Universidade	Rensselaer Polytechnic Institute

Logo	
Acrônimo	FSMT
Nome	*Feature* Solid Modeling Tool
Homepage	
Principal Pesquisador	J. Zhou
Universidade	Huazhong University of Science and Technology

Logo	
Acrônimo	QTC
Nome	Quick Turnaround Cell
Homepage	http://www.cadlab.ecn.purdue.edu/qtc/
Principal Pesquisador	David. Anderson
Universidade	Purdue University

Logo	
Acrônimo	*FESTEVAL*
Nome	*Feature* Based Support for the Development Process Chain Design - Planning - Manufacturing'
Homepage	http://www.unimep.br/feau/scpm/festeval.htm
Principal Pesquisador	Klaus Schützer
Universidade	Universidade Metodista de Piracicaba
Sistema CAD	Unigraphics
Estilo	Projeto
Arquivo	STEP - AP 224
Ano	2000
Linguagem	C++