UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UMA CONTRIBUIÇÃO PARA A INTEGRAÇÃO ENTRE SISTEMAS CAE E CAD COM O DESENVOLVIMENTO DE UM APLICATIVO

ENG. JEOVANO DE JESUS ALVES DE LIMA

ORIENTADOR: PROF. DR.-ING. KLAUS SCHÜTZER

Santa Bárbara D'Oeste

2010

UNIVERSIDADE METODISTA DE PIRACICABA

FACULDADE DE ENGENHARIA, ARQUITETURA E URBANISMO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE PRODUÇÃO

UMA CONTRIBUIÇÃO PARA A INTEGRAÇÃO ENTRE SISTEMAS CAE/CAD COM O DESENVOLVIMENTO DE UM APLICATIVO

ENG. JEOVANO DE JESUS ALVES DE LIMA

ORIENTADOR: PROF. DR.-ING. KLAUS SCHÜTZER

Dissertação de Mestrado apresentada no 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

2010

Agradecimento

A Deus, em primeiro lugar.

Ao Professor Dr.-Ing. Klaus Schützer pela compreensão, orientação e incentivo dados a mim para a conclusão deste trabalho.

Aos companheiros de trabalho do Laboratório de Sistemas Computacionais para Projeto e Manufatura pelo apoio e incentivo.

A CAPES – Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, pela concessão da bolsa de estudos.

E a todos que de alguma forma colaboraram para que eu chegasse até aqui, em especial a minha querida esposa Mônica por ter me apoiado mesmo nos momentos mais difíceis e ao nosso filho Samuel.

“Adquire a sabedoria, adquire o entendimento e não te esqueças das palavras da minha boca, nem delas te apartes.”
(Pv. 4,5)

Sumário

Lista de Figuras

Lista de Siglas

ANSI – American National Standard Institute

API – Application Programming Interface

BRep – Boundary Representation

CAD – Computer Aided Design

CAE – Computer Aided Engineering

CAPP – Computer Aided Process Planning

CAM – Computer Aided Manufacturing

CAx – Computer Aided x (qualquer sistema auxiliado por computador)

CFD – Computational Fluid Dynamics

CSG – Constructive Solid Geometry

DES – Data Encryption Standard

DIN – Deutsches Institut für Normung

DLL – Dynamic Library

FEM – Finite Element Method

IDE – Integrated Development Environment

IGES – Initial Graphics Exchange Specification

ISO – International Organization for Standardization

ODE – Ordinary Differential Equation

PDM – Product Data Management

STEP – Standard for the Exchange of Product Model Data

XML – eXtensible Markup Language

Resumo

Lima, Jeovano de Jesus Alves de. Uma Contribuição para a Integração entre Sistemas CAE e CAD – Desenvolvimento de um Aplicativo para os Sistemas MDESIGN e SIEMENS NX. 2010. 113 f. Dissertação (Mestrado em Engenharia de Produção) – Faculdade de Engenharia, Arquitetura e Urbanismo, Universidade Metodista de Piracicaba, Santa Bárbara d'Oeste.

As empresas procuram cada vez mais se adaptar às novas tendências visando à competitividade imposta pelo mercado. Um dos importantes fatores de diferenciação entre as empresas está na utilização de ferramentas computacionais como as tecnologias CAD/CAE. Com o devido conhecimento em tais ferramentas pode-se alcançar melhorias no processo de produção devido ao ganho de tempo e facilidade nas adaptações necessárias durante o ciclo de desenvolvimento do projeto. Entretanto, a troca e manipulação dos dados entre os sistemas CAD/CAE vêm gerando diversas dificuldades devido à grande diferença no entendimento dos mesmos por cada sistema, e pela necessidade de transmitir uma grande quantidade de dados. Este trabalho teve por finalidade desenvolver um aplicativo para integração entre uma ferramenta de dimensionamento de componentes mecânicos com o sistema CAD, para geração automática do modelo 3D, a partir dos resultados do dimensionamento, visando à diminuição de tempo e a eliminação dos erros na manipulação dos dados.

Palavras chave: Desenvolvimento do Produto, Integração CAE/CAD, XML.

Abstract

Lima, Jeovano de Jesus Alves de. Uma Contribuição para a Integração entre Sistemas CAE e CAD – Desenvolvimento de um Aplicativo para os Sistemas MDESIGN e SIEMENS NX. 2010. 113 f. Dissertação (Mestrado em Engenharia de Produção) – Faculdade de Engenharia, Arquitetura e Urbanismo, Universidade Metodista de Piracicaba, Santa Bárbara d'Oeste.

Companies increasingly seek to adapt to new trends in order to competition imposed by the market. One of the most important differentiating factors between companies is the use of computer technologies as CAD/CAE. With the knowledge of those tools it's possible to achieve improvements in the production process due to time saving and facility of the necessary adjustments during the cycle of project development. However, exchange and manipulation of data between CAD/CAE systems have generated several difficulties due to the large difference in the uptake by each system, and the necessity of transmitting large amounts of data. This study aimed to develop an application for an integration of a new tool for design of mechanical components with the CAD system, for automatic generation of 3D model from the results of scaling in order to reduce time and eliminating errors in data manipulation.

Keywords: Product Development, Integrating CAE / CAD, XML.

1 Introdução

Em busca da competitividade, as empresas procuram cada vez mais se adaptar às novas tendências do mercado, que exigem uma grande diversificação dos produtos. Com isso, a velocidade de execução e implementação de um projeto é fundamental para o sucesso de uma empresa. As tecnologias CAD/CAE são essenciais para que essas exigências no lançamento de novos produtos sejam atingidas, além de desempenharem um papel fundamental para a viabilização de um projeto de produto em tempo reduzido, oferecendo oportunidade para simular e reduzir custos na fase de seu desenvolvimento [1].

Neste cenário, a criatividade é um fator importante no desenvolvimento de novos produtos, aliada ao uso de ferramentas de projeto avançadas, conceitos de engenharia reversa, projeto enxuto e ao uso de métodos inovadores de fabricação. O sucesso e a permanência da indústria neste mercado competitivo estão diretamente ligados à sua capacidade de introduzir novos produtos em menor tempo e com baixo custo, aumentando a qualidade e o valor tecnológico.

Neste contexto, os sistemas computacionais (CAx) estão mais presentes no desenvolvimento de projetos dentro das indústrias. Com o auxílio deles é possível obter uma otimização estrutural e geométrica do produto final com uma redução significativa no custo do projeto, fazendo com que o tempo de desenvolvimento seja bem menor quando comparado com outros processos convencionais de projeto e construção [2].

Segundo Cunha [3] muitos autores procuraram compreender e configurar um passo a passo no projeto do produto para entender a importância de se utilizar sistemas computacionais.

De acordo com Pahl e Beitz [4] o processo de projeto do produto pode ser dividido em quatro etapas: clarificação das necessidades, projeto conceitual, projeto preliminar e projeto detalhado – estas podem ser vistas na Figura 1.1, que mostra também a utilização dos sistemas computacionais CAE para projeto conceitual/preliminar e CAD para projeto Preliminar/Detalhado.

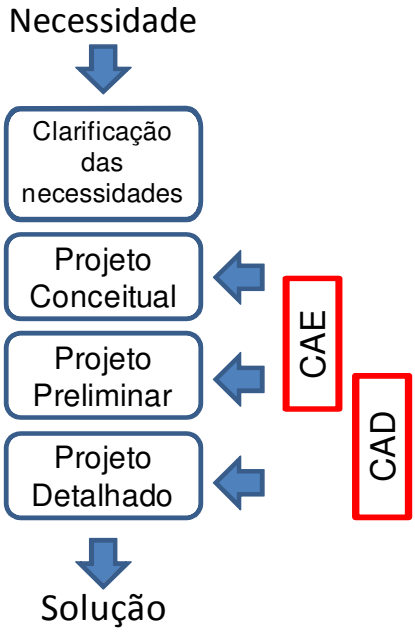


Figura 1.1: Fases do projeto [4].

A primeira fase é o Planejamento e clarificação das necessidades. Nesta etapa se analisa a situação do mercado, da empresa e da conjuntura. Serão levantadas as necessidades a serem atendidas, venham elas como pedidos de desenvolvimento, sob a forma de um produto por parte do setor de planejamento de produtos, como pedidos de um cliente ou mesmo como críticas ou sugestões. Ao final desta etapa, busca-se elaborar uma lista detalhada de requisitos que contenha as restrições e os objetivos a serem alcançados, além de uma descrição do desejo ou exigência requerido pelo cliente [5].

A segunda fase seria o Projeto Conceitual, porem é considerada a fase inicial do processo de projeto do produto. Essa fase do projeto, assim como as outras, deve ser sistematizada, visando à aplicação do computador para que seja possibilitada sua integração com as demais fases do projeto do produto [6].

No Projeto Conceitual, informações dispersas são manipuladas e tratadas, muitas oriundas do raciocínio do próprio projetista e, portanto, não formalizadas [7]. Devido a isso, há a necessidade de vasto uso de criatividade por parte do projetista, o que, significa que há grande influência dessa capacidade criadora na qualidade final do projeto.

Deve-se realçar que é grande a importância da fase inicial de projeto no custo e no sucesso do produto final, conforme mostrado na Figura 1.2. Decisões tomadas nessa fase apresentam grande dificuldade e proporcionam alto custo para serem alteradas nas fases posteriores do desenvolvimento de um produto [8].

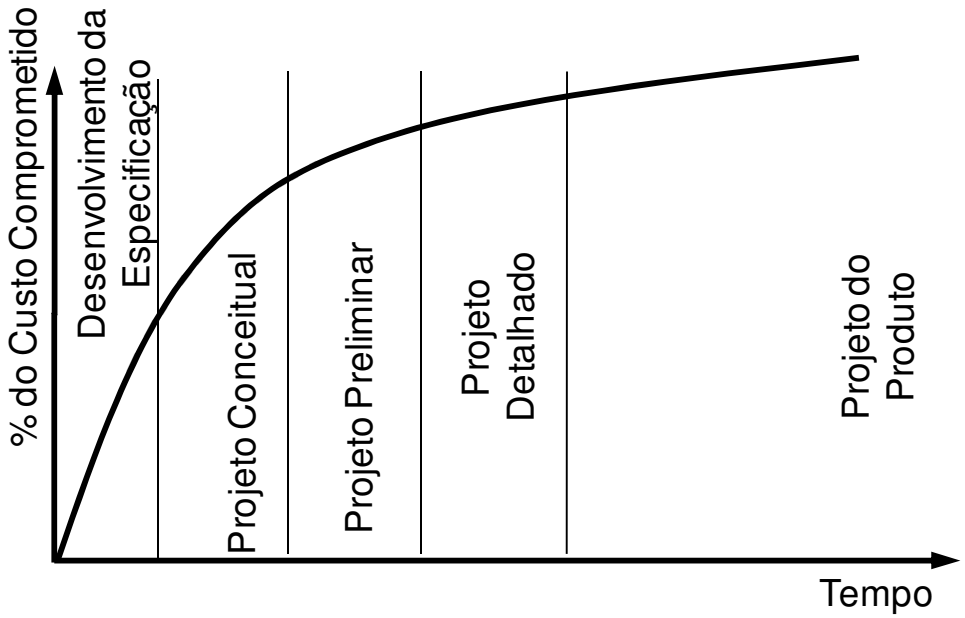


Figura 1.2: Influência das fases do projeto no custo do produto [8].

Tendo-se a concepção inicial, pode-se então passar para a fase de Projeto Preliminar, no qual essa concepção prossegue até que se encontre um projeto preliminar para o produto. É um processo complexo, no qual muitas atividades são executadas simultaneamente, utilizando processos iterativos. Alguns pontos devem ser abordados e alcançados nesta etapa sendo alguns deles: identificação de requisitos determinantes, a partir da lista de requisitos, formalizada na primeira etapa da metodologia; avaliação de anteprojetos preliminares por meio de critérios preestabelecidos, de acordo com o julgamento do projetista, selecionando um ou mais de um; detalhamento dos anteprojetos selecionados, avaliando também sua viabilidade técnica e econômica, chegando-se assim, ao projeto básico global preliminar. Por fim, deve-se elaborar a lista de peças preliminares e as instruções preliminares para produção e montagem [5].

Na última etapa, o projeto detalhado como o próprio nome diz, é a fase em que é feito o detalhamento de todos os dados obtidos na fase conceitual e preliminar. Esses dados serão os utilizados para as configurações finais do produto, desenhos com tolerâncias, planos de processo/fabricação/montagem, BOM, etc. Algumas das ferramentas computacionais utilizadas nesta fase são: CAD, CAM, CAE [9].

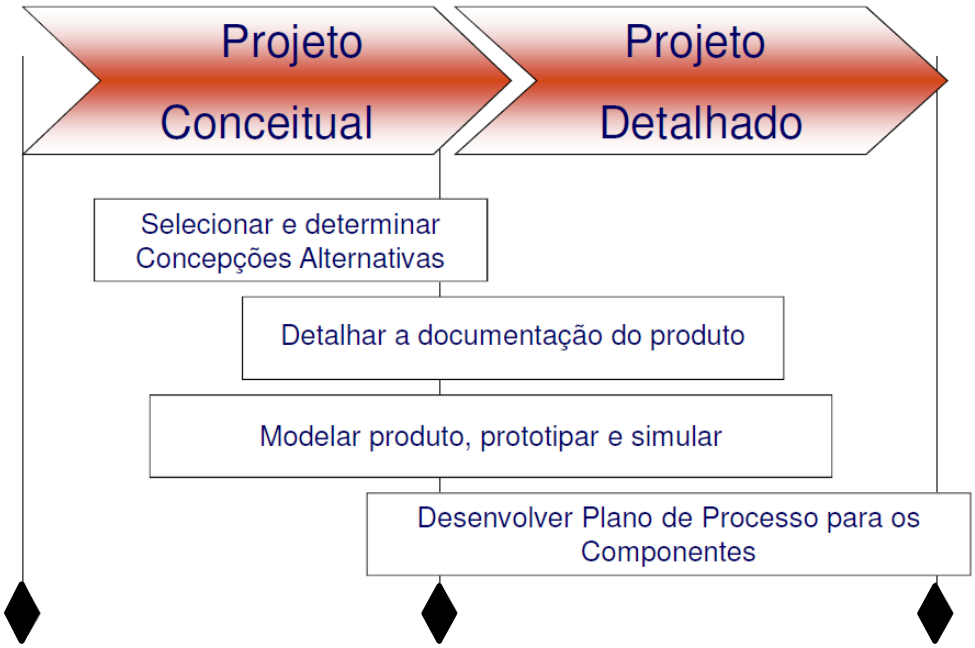


Figura 1.3: Sobreposição das fases do desenvolvimento do projeto [9].

Essa fase em alguns momentos no ciclo de desenvolvimento se sobrepõe à fase do projeto conceitual conforme mostrado na Figura 1.3. Isso torna essencial a integração das ferramentas computacionais utilizadas nessas fases, buscando ganho de tempo e minimização de erros decorrentes da manipulação manual das informações [9].

Ao longo do ciclo de desenvolvimento do produto, muitas informações são geradas para permitir que ele seja produzido e chegue às mãos dos clientes com qualidade e rapidez. Daí a importância da integração de ferramentas computacionais envolvidas no desenvolvimento do produto [10].

Visando colaborar para o ciclo de desenvolvimento de produto, especificamente nas fases de projeto, este trabalho propõe um conceito de integração entre os sistemas CAE e CAD com o desenvolvimento de um aplicativo para integração entre uma ferramenta de dimensionamento de componentes mecânicos e um sistema CAD, visando à geração automática do

modelo 3D, a partir dos resultados do dimensionamento (XML), a diminuição de tempo e a eliminação dos erros na manipulação dos dados.

1.1 Estrutura do trabalho

A estrutura deste trabalho é composta de seis capítulos que são apresentados a seguir com uma breve abordagem do respectivo conteúdo destes.

Capitulo 1 – Introdução.

Capítulo 2 – Estado da Arte dos Sistemas CAE/CAD e sua Integração – Revisão bibliográfica sobre os sistemas CAE/CAD, integração entre estes sistemas, linguagem XML, construção de API utilizando interfaces de programação de sistemas CAD.

Capítulo 3 – Proposta do Trabalho – Apresentação dos objetivos deste trabalho e a metodologia que determina como será desenvolvido o aplicativo para integração e avaliação da interface com usuário.

Capítulo 4 – Desenvolvimento e Implementação do Aplicativo – Neste capítulo, será apresentado o modelo conceitual do aplicativo, como foram feitas as implementação da interface de integração e a avaliação da mesma.

Capítulo 5 – Conclusões e sugestões para trabalhos futuros.

Capítulo 6 – Bibliografia Referenciada – Lista das referências utilizadas durante o desenvolvimento deste trabalho.

2 Estado da Arte dos Sistemas CAE e CAD e sua Integração

Este capítulo apresenta um histórico dos sistemas CAE e CAD, a necessidade de comunicação entre estes sistemas, bem como a importância da utilização destes no desenvolvimento do projeto e da utilização dos dados gerados pelos sistemas CAE como base na integração entre eles, possibilitando com isso reduzir o *time to market* do produto.

2.1 Sistemas CAE

Diariamente, os engenheiros e os projetistas se defrontam com problemas técnicos – alguns simples e outros mais complexos – tendo que solucioná-los. Para isso, utilizam diversas fontes de informações e recursos como fórmulas, tabelas e todo conhecimento que aprenderam nos cursos de engenharia, bem como o auxilio dos computadores, daí o surgimento dos sistemas CAE [11].

Computer Aided Engineering (Engenharia Auxiliada por Computador) são sistemas que permitem realizar uma grande quantidade de cálculos voltada para o dimensionamento e otimização de diversas estruturas e componentes mecânicos.

De forma a minimizar os esforços do engenheiro, possibilitam uma maior concentração nas atividades de projeto com mais criatividade, preocupando-se menos com a parte operacional e mais com a questão estratégica, fazendo do CAE uma ferramenta poderosa para redução de custos de um projeto, minimizando o tempo para o lançamento de um produto [12].

Para REHG, “CAE é a análise e resolução de projetos de engenharia usando técnicas computacionais para calcular parâmetros operacionais, funcionais e de manufatura do produto, que são muito complexos para os métodos tradicionais” [13].

Já VAJPAYEE afirma que “CAE é um termo genérico que abrange tarefas essenciais na engenharia de um produto utilizando o computador. Isto envolve seleção de material, análises de resistência, vibração, ruído, deformação térmica e assim sucessivamente. Para assegurar a qualidade do produto, as análises de resistência são amplamente utilizadas” [14].

Há diversas ferramentas CAE no mercado que utilizam métodos analíticos, de equações ordinárias diferenciais (ODE), normas e outros materiais para realização de cálculos padrões. Esses métodos são mais simples e rápidos, pois não precisam de modelo 3D.

Segundo Garcia [15] e Almeida [16] na engenharia do produto há basicamente três caminhos para se resolver um problema: o método analítico, o método experimental e os métodos numéricos. O método analítico é aquele baseado em equações desenvolvidas com grande base teórica para certas situações. Para sua aplicação, deve-se proceder a uma modelagem matemática da realidade. Sua vantagem é a exatidão da resposta para aquela modelagem e a exigência de apenas lápis e papel.

A solução analítica geralmente só é disponível para problemas simples e sua aplicação na maioria das vezes exige hipóteses de difícil ocorrência na prática, tais como homogeneidade das características do material, isotropia e linearidade de resposta, o que a torna distante da realidade.

O método experimental apresenta algumas vantagens interessantes, como a alta confiança na resposta obtida e a simplificação de cálculos. Porém, trás também fortes desvantagens, como o alto custo da confecção de protótipos, a validade da resposta para casos específicos, sendo necessários novos testes de validação para qualquer modificação de projeto e o tempo consumido, pois qualquer alteração da forma do modelo exige a confecção de um novo, com os custos relativos.

Os métodos numéricos são métodos de convergência que apresentam uma seqüência de cálculos simples, porém repetitivos. Estes métodos simulam uma realidade e apresentam vantagens como possibilidade de executar várias versões de possíveis soluções a fim de otimizar-se a resposta com rapidez, trazendo consigo menor custo em relação aos métodos experimentais e razoável facilidade de execução. Entre esses métodos, destaca-se o Método dos Elementos Finitos, um método com grande disseminação na área da engenharia do produto, em empresas de projeto [17].

Este método aplica-se às mais diversas áreas da engenharia, tais como análise estrutural, problemas de contato, grandes deformações, fluxo de calor ou material, ligações e uniões, ou quaisquer problemas que possam ser expressos através de equações diferencias. Este método tem sido cada vez mais utilizado

na prática, devido as suas características já citadas. Atualmente, é grande o número de softwares comerciais disponíveis, porém são necessários profissionais altamente qualificados para utilizá-los.

Os sistemas CAE (*Computer Aided Engineering*) utilizam para a resolução de problemas complexos de engenharia os modelos matemáticos (métodos numéricos) como, por exemplo, o Método dos Elementos Finitos (FEM), elementos de contorno, diferenças finitas e volumes finitos. Esses métodos otimizam as resoluções de problemas de equilíbrio (concentração de tensões, análise de tensões em pistões e materiais compostos), determinação de autovalores (freqüências naturais em máquinas-ferramentas) e de propagação (fraturas e rupturas sob cargas dinâmicas) [16]. Entretanto, alguns cálculos da Engenharia Mecânica não requerem uma análise tão complexa e muitas vezes a utilização do FEM não se justifica:

- Grande parte dos componentes mecânicos, como por exemplo, (Eixos, engrenagens, rolamentos, mancais, etc.), que já possuem soluções analíticas, normas desenvolvidas ao longo dos anos;
- O custo computacional para sistemas CAE, que utilizam o Método dos Elementos Finitos é muito alto, necessitando de um hardware com maior capacidade de processamento e memória;
- Para utilização de sistemas baseados no Método dos Elementos Finitos são necessários profissionais altamente qualificados.

2.1.1 Método dos Elementos Finitos (FEM)

O Método dos Elementos Finitos (FEM - *Finite Element Method)* teve suas origens na análise estrutural. Com o surgimento dos primeiros computadores, no início da década de 50, os métodos matriciais para a análise estrutural tiveram um grande desenvolvimento. A crescente demanda por estruturas mais leves, tais como as encontradas na indústria aeronáutica, conduziu ao desenvolvimento de métodos numéricos que pudessem ser utilizados nas análises de problemas mais complexos [18].

Na década de 70 o FEM teve suas aplicações estendidas a problemas de mecânica dos fluidos e, desde então, vem consolidando-se como um método mais geral de solução de equações diferenciais [17]. As equações diferenciais descrevem diversos fenômenos físicos que são de interesse na engenharia,

como por exemplo, de transmissão de calor, deformação de estruturas, escoamento de fluidos, etc.

O método dos elementos Finitos utiliza o modelo 3D (Superfície ou volume) como base para análise, esta análise e feita a partir de pequenas partes chamadas elementos e nós que ligam estes elementos, formando uma malha conforme mostrado na Figura 2.1, além do modelo são utilizadas um conjunto de informações como carregamento, condições de contorno e métodos de resistências de materiais estão diretamente ligados para resolução obtenção dos resultados desejados [19].

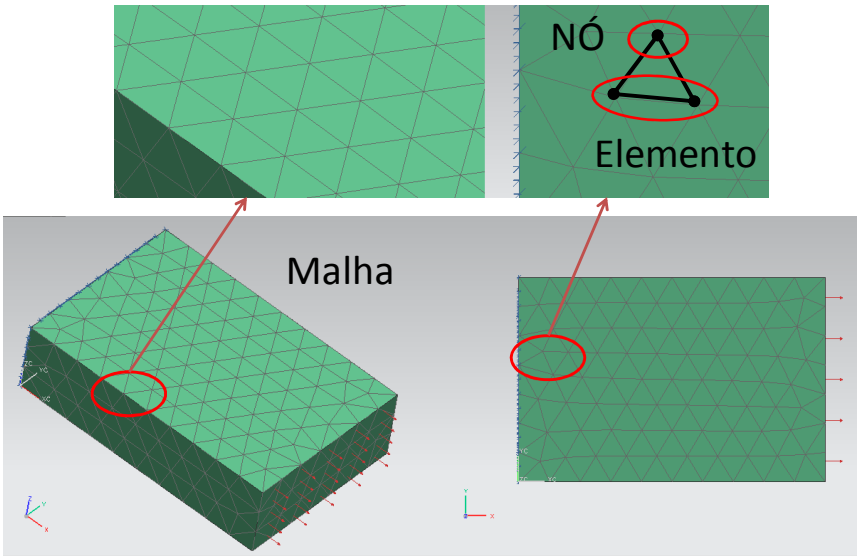


Figura 2.1: Exemplo elementos, nó e malha

O desenvolvimento de projetos baseados no Método de Elementos Finitos (FEM), devido à sua rapidez e baixo custo, vem sendo cada vez mais utilizado nas várias áreas da engenharia [15]. Os programas computacionais baseados no FEM se estruturam em quatro etapas principais, conforme Figura 2.2:

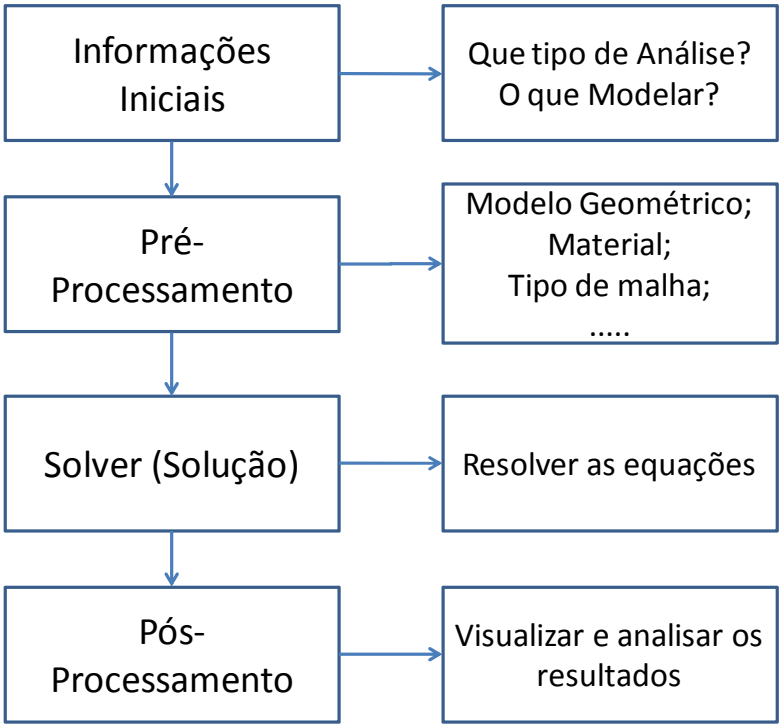


Figura 2.2: Quatro etapas principais do Método dos Elementos Finitos [15]

No pré-processamento, o modelamento do produto é fornecido por um sistema CAD. Caso não estejam integrados os sistemas CAD/CAE (sistemas de fabricantes diferentes), a geometria deverá ser exportada pelo CAD por meio de uma interface padrão (IGES, STEP), importada no sistema CAE e feita à geração da malha, que é fundamental para a qualidade da simulação, pois os nós entre os elementos são utilizados como base para todos os cálculos de solicitação realizados no CAE [11].

Ainda no pré-processamento, todas as condições de contorno devem ser fornecidas, tais como: região de solicitação mecânica, resistência do material do produto, temperatura de trabalho, dentre outras informações que comporão a simulação [19].

O *solver* é responsável por fazer os cálculos, utilizando algoritmos matemáticos. O pós-processamento é responsável pela visualização dos resultados, que normalmente são apresentados em forma de gráficos.

Segundo Almeida [17] a solução por FEM vem ganhando terreno na área da engenharia do produto. Apesar de ser uma solução aproximada, cuja exatidão depende muito da experiência do projetista, suas características são de relativo baixo custo, alta flexibilidade e disponibilidade para a resolução de praticamente qualquer tipo de problema, conforme mostrado a seguir:

- Análise estática: nos problemas estruturais estáticos define-se a geometria da estrutura que será analisada assim como os materiais, suas propriedades, as cargas e os apoios que irão compor a estrutura a ser analisada. As cargas são aplicadas na estrutura de modo a não causarem vibrações na mesma e seus deslocamentos são considerados pequenos. O principal objetivo desse tipo de problema é calcular as tensões e deformações ou deslocamentos da estrutura (Figura 2.3) [10, 17].

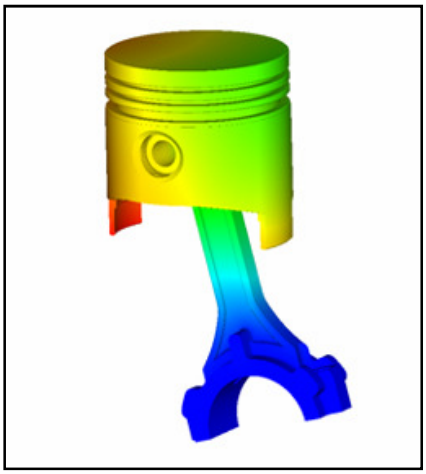


Figura 2.3: Exemplo FEM análise estática

- Análise dinâmica transiente: o princípio desse tipo de problema é o mesmo dos problemas estruturais estáticos, mas com a diferença de que nesse caso as cargas são aplicadas dinamicamente, causando assim vibrações na estrutura que está sendo analisada. Nesse tipo de problema todas as forças inerciais devem ser consideradas e o objetivo principal é calcular tensões, deslocamentos, velocidades e acelerações que podem ser ocasionados na estrutura (Figura 2.4) [17].

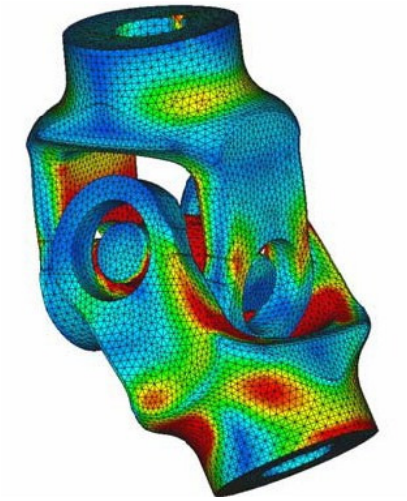


Figura 2.4: Exemplo FEM análise dinâmica

- Análise de freqüência natural (Modal): registra a tensão provocada e causada por vibração sonora nas freqüências naturais. Por exemplo, a força destrutiva das vibrações criadas por terremotos pode ser aplicada a construções e pontes para determinar os limites no intuito de evitar falhas catastróficas. Aplicadas também em motores e compressores [10].
- Análise de movimento: analisa questões de velocidade, aceleração e deslocamento para produtos que possuem essas características. Além disso, outros parâmetros podem ser especificados, como limite de movimento de um determinado componente e interferência. Como exemplo, pode-se determinar o grau de inclinação e o tamanho da barra metálica que interliga as partes de uma janela basculante, para que haja o mínimo de esforço possível no momento de sua abertura [15].
- Análise de Fluidos: na análise de fluídos (CFD - *Computational Fluid Dynamics*) podem-se considerar gases e líquidos, e ela pode ser feita em movimento ou não. Seu principal objetivo é conseguir determinar quais as forças induzidas pelos fluidos em estruturas comuns de engenharia [20, 21].

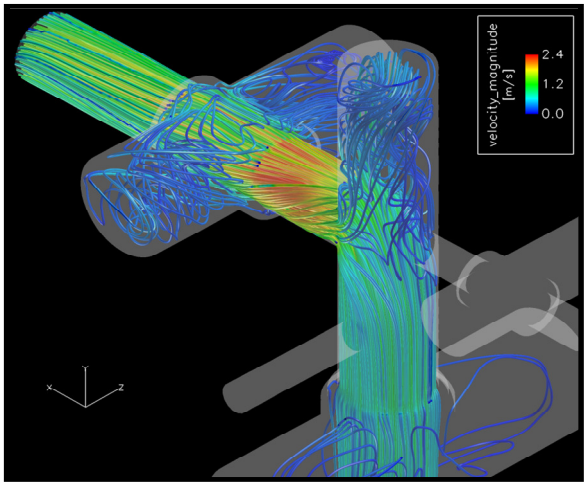


Figura 2.5: Exemplo CFD análise de fluxo

- Análise Térmica: a análise térmica tem com objetivo determinar as tensões e as deformações de um corpo quando tem sua temperatura alterada. Através dessa análise, deseja-se determinar a distribuição das temperaturas em um corpo quando o mesmo é exposto a uma fonte de calor externa. A distribuição da temperatura do corpo leva em consideração as propriedades térmicas do material [20, 21].
- Problemas Eletro-Eletrônicos: utilizados na indústria de eletro-eletrônicos e baseiam-se nas regras da eletrônica para conseguir determinar a diferença de potencial, campos eletromagnéticos, indutância, torque e força em motores elétricos, transformadores e reatores. Também é realizada a análise térmica dos materiais bipolares como circuitos eletrônicos.
- Problemas de Contato e Grandes Deformações: nesse tipo de problema, levam-se em consideração principalmente as condições de contato entre sólidos elásticos e as propriedades de deformação do material, que são consideradas deformações grandes, tornando a análise não-linear. Esse tipo de análise é utilizada para simular processos de estamparia, de conformação, processos metalúrgicos, etc.

2.1.1.1 Vantagens do FEM

A seguir seguem algumas vantagens de utilizar os sistemas CAE do tipo FEM no desenvolvimento do produto:

- Maior eficiência: trabalhos realizados com o CAE demandam muito menos tempo, pois os cálculos e análises referentes ao projeto são rapidamente feitos por métodos numérico-computacionais;
- Engenharia Preditiva: a utilização de uma abordagem de engenharia preditiva ataca problemas já no ciclo de projeto, reduzindo o número de correções de alto custo na fase de implementação [17, 20];
- Menor Custo: a simulação de protótipos virtuais é muito mais barata em relação a um protótipo real e em comparação com as técnicas tradicionais de engenharia, baseadas na confecção de protótipos muitas vezes caros e que consomem uma grande quantidade de tempo [20, 21];
- Aprimoramento do Produto: as simulações virtuais permitem rapidamente avaliar o desempenho do projeto com o meio em que irá atuar [20, 21];
- Agilidade e Flexibilidade da Empresa no Mercado: as rápidas transições que acompanham o Ciclo de Vida do Produto necessitam de agilidade e flexibilidade do setor de Engenharia para empreender novos projetos. Isto é proporcionado pelo CAE, devido à sua maior eficiência [22];
- Reutilização: tanto cálculos como desenhos de partes do projeto são reaproveitados em novos projetos, reduzindo o tempo do desenvolvimento do projeto [15];
- Rápida Alteração: ligada à reutilização, algumas vezes é preciso apenas redimensionar algum componente ou material, como alterar o valor do capacitor no projeto, por exemplo [15].

2.1.1.2 Desvantagens do FEM

Do outro lado, as desvantagens estão basicamente relacionadas com [18 a 20]:

- Necessidade de estações de trabalho dedicadas para realização dos cálculos complexos;
- Necessidade de um especialista para preparar o modelo e interpretar os resultados processados pelos sistemas de CAE;
- Necessidade da utilização de protótipos, pois não é possível realizar todas as simulações necessárias a fim de assegurar a qualidade do produto final através do software;
- Alto custo do CAE, dependendo de sua finalidade;
- Problemas de integração com outras aplicações, como por exemplo, o CAM.

2.1.2 Dimensionamento Analítico

Quando se fala de dimensionamento analítico a intenção na verdade é referir-se a formulações que proporcionam o desenvolvimento de componentes mecânicos ainda em sua fase conceitual. Nesta fase, não há modelos geométricos para verificação que emprega métodos de elementos finitos, mas isso não é uma regra e, por isso, utiliza-se de todos os meios de informações e materiais de apoio para se conseguir resultados tão reais quanto possíveis.

Segundo Niemann [23] e Norton [24], os fatores mensuráveis são relevantes para cálculos cujo intuito é prever ou verificar os custos da construção (dimensão e pesos), os efeitos a serem obtidos e a vantagem da solução no desenvolvimento do projeto de elemento de máquinas. Alguns exemplos de fatores mensuráveis são: Peso, resistência, durabilidade, desgaste, deformação, flexibilidade, atrito, potência, dimensão, segurança, consumo de energia, etc. Esses fatores influenciam a configuração do sistema, o processo de fabricação e montagem, a definição das dimensões e o material utilizado.

Dependendo do projeto, existem particularidades que precisam ser levadas em consideração e observadas com cuidado para não cometer erros grosseiros. O emprego de tabelas e gráficos de valores contribui para a sua precisão. Entretanto, é necessário que, ao calcular, tenha-se idéia do significado das equações e resultados, pois somente compreendendo o que se está fazendo é possível evitar erros e chegar ao resultado esperado [25].

Assim sendo, os projetos de elementos de máquinas, que segundo Spotts e Shoup [26] são considerados como um subconjunto do projeto mecânico, responsáveis pelo dimensionamento e transmissão de movimentos de um produto, como um projeto de redutores de velocidades, máquinas de levantamento, pontes rolantes ou máquinas operatrizes, por exemplo.

O papel dos elementos de máquinas no ciclo de desenvolvimento do projeto é colocado de forma objetiva e segue um domínio hierárquico. Uma forma de pensar sobre isso é considerar vários círculos ou sub-áreas [26], ilustrada na Figura 2.6.

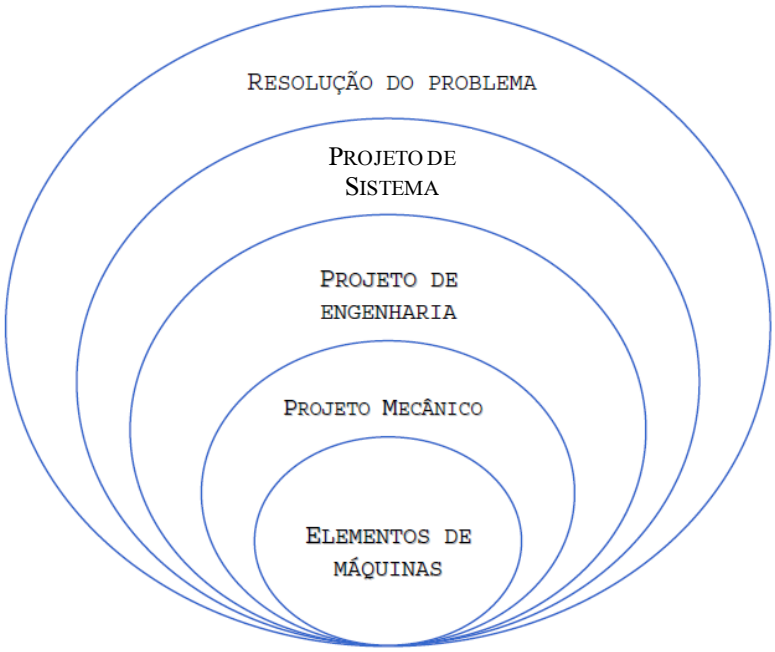


Figura 2.6: Hierarquia de projetos de elementos de máquinas [27]

Segundo Silveira, [27] vários programas isolados para dimensionamento de alguns componentes mecânicos vêm sendo desenvolvidos nos últimos anos, sejam através de meios acadêmicos ou comerciais (Figura 2.7). Alguns são genéricos, pois possibilitam a automatização de qualquer procedimento de cálculo através de modelamento matemático natural, escrevendo as fórmulas matemáticas da mesma forma que são encontradas nos livros (Mathcad), ou através de programação, definindo variáveis para em seguida descrever as formulas (Matlab), ou também através de planilhas que possuem células em que podem ser inseridas variáveis, funções matemáticas e condições de contorno (Excel). Já outros são mais específicos, pois são dedicados a determinados tipos de processos, como é o caso do MDESIGN, que constitui-

se em um sistema CAE para dimensionamento de elementos mecânicos padrões.

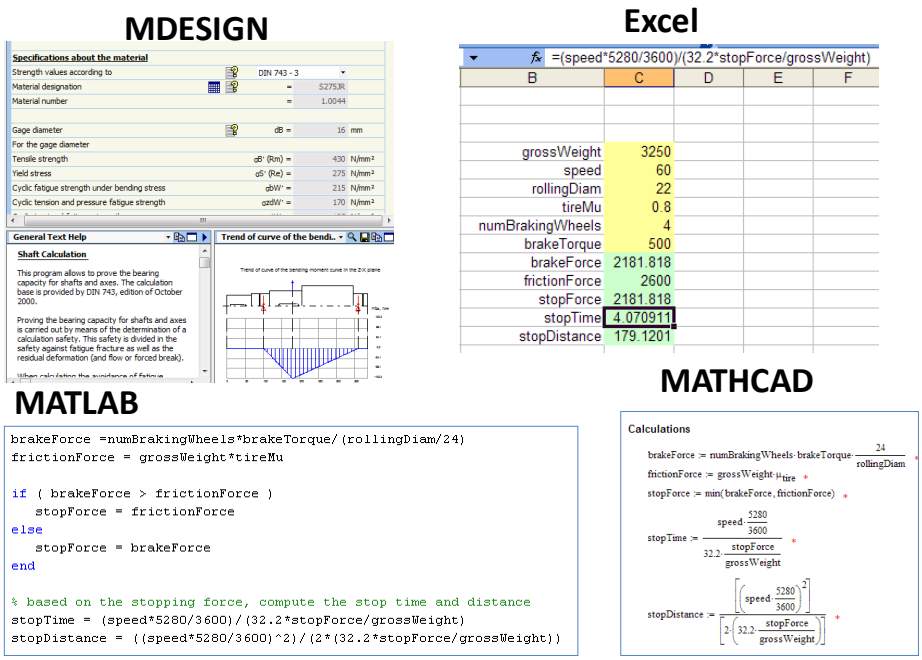


Figura 2.7: Sistemas CAE de dimensionamento Analítico

Na metodologia tradicional de projeto, os engenheiros, partindo de uma idéia, iniciam o dimensionamento dos componentes de um produto e recorrem a uma série de informações provenientes de manuais, normas técnicas, métodos de cálculo e, provavelmente, uma calculadora para auxiliá-los nos cálculos. O modelamento do produto é resultado do processo interativo entre cálculo e análise, seguido da construção de protótipos. A Figura 2.8 apresenta essa metodologia [27].

Figura 2.8: Sequência tradicional de dimensionamento de elementos mecânicos.

O ciclo tradicional do projeto dificulta o desenvolvimento de alternativas para projeto, devido à limitação do tempo despedido nas atividades manuais de projeto (modelamento). Segundo Schützer [28] e Besant [29], o primeiro software de auxílio à engenharia foi o CAD (*Computer Aided Design*), uma vez que esta etapa consumia um tempo maior no desenvolvimento do projeto.

Posteriormente, foram desenvolvidos outros sistemas de apoio à engenharia, como por exemplo, os sistemas: CAE, CAM e CAPP. O objetivo destas ferramentas era a redução do ciclo de desenvolvimento do produto pela integração de suas etapas. A utilização do computador nas diversas áreas ligadas ao produto e especificamente no projeto gerou uma evolução na maneira de projetar [28].

2.2 Sistemas CAD

O sistema CAD (*Computer Aided Design*) teve origem em projetos desenvolvidos no Instituto de Tecnologia de Massachusetts (MIT - *Massachusetts Institute of Technology*), que datam de 1956. Já o primeiro sistema que permitia que o usuário interagisse graficamente com o computador foi o *Sketchpad* desenvolvido pelo MIT em 1963, servindo tanto para a execução de cálculos quanto para a visualização gráfica do projeto [30, 31].

Nos últimos anos, os sistemas CAD tiveram um grande avanço, onde muitas empresas adotaram esta tecnologia nos seus processos de produção e deixaram de utilizar pranchetas de desenho [32]. A introdução do sistema CAD

no processo de projeto convencional trouxe uma grande evolução no desenvolvimento, análise, otimização e modificação do modelo [33], permitindo que outros tipos de dados pudessem ser introduzidos ao projeto ou mesmo reconhecidos automaticamente pelo sistema, fazendo-se necessária a integração deste com todas as outras fases do projeto. Essa integração ocorreu quando o CAD interagiu com outros sistemas como o CAE (*Computer Aided Engineering*), CAM (*Computer Aided Manufacturing*) e o CAPP (*Computer Aided Process Plan*).

O principal objetivo do sistema CAD nos dias atuais é auxiliar o projetista no modelamento de peças, utilizando a interação com o computador para definir todas as informações geométricas que serão utilizadas posteriormente pelos outros sistemas CAx. Estas informações são armazenadas, podendo ser modificadas futuramente. Há várias maneiras de construir-se um modelo, tanto em modo 2D quanto 3D, usando entidades geométricas primitivas que são introduzidas ao sistema por meio de símbolos gráficos ou linguagens de programação textuais.

Segundo Valentin e Correia [34], o modelo geométrico tem sido parte integrante da indústria desde a sua concepção como organização produtiva, pois o modelo geométrico é o elo entre o departamento de projeto e a produção.

O projeto preparado com padrões pré-determinados faz com que a informação seja rapidamente comunicada para o resto da fábrica, proporcionando a confecção do produto idealizado com maior rapidez. Por esse motivo, aliado à grande evolução do poder de processamento e a queda dos preços dos computadores, o número de profissionais que utilizam o CAD como ferramenta de trabalho vem aumentando progressivamente [35]. Por isso, Foggiato *et al*. [36] ressalta que embora os sistemas CAD estejam estabelecendo-se como padrão na maioria das empresas, há uma carência de informações sobre a maneira correta de se gerar os modelos. Isto tem levado muitos profissionais da área a produzir modelos tridimensionais sem nenhum critério, o que resulta em perda de tempo na reutilização dos mesmos por outros projetistas.

2.2.1 Modelamento no Sistema CAD

O modelamento 3D apresenta as dificuldades que são próprias do processo de projetar. O projetista é obrigado a considerar as três dimensões

simultaneamente e, em alguns casos, a utilização do modelo 3D é imprescindível, como na aplicação de análises por elementos finitos para verificação de tensões, escoamento, temperatura, etc. e ainda quando há a necessidade de calcular-se o volume, propriedades de massa, o eixo de inércia e verificação de interferências [34].

A seguir são apresentados os principais métodos de representação 3D [35].

- Wireframe;
- Superfície;
- CSG;
- BRep,
- Baseado em *Features;*
- Paramétrico.

Modelamento por *Wireframe*: A modelagem por *wireframe* (armação em arame) baseia-se na união de linhas entre pontos no espaço 3D, criando modelos espaciais. Não há superfícies em um modelo *wireframe*, só vértices e linhas retas e curvas, que representam as arestas de um objeto 3D. Este tipo de modelagem emprega menos processamento do que as outras, porém, segundo McMahon e Browne [37], não é completa, nem livre de ambigüidades, nem possível de determinar as seções, volume ou massa do objeto a partir do modelo. Apesar dos modeladores *wireframe* terem sido os pioneiros, não é mais utilizado [34, 35].

Modelamento por Superfície: A modelagem por superfícies é indicada quando o modelo possui curvas complexas ou superfícies livres. Zeid [38] confirma sua aplicação para a representação de objetos com geometrias complexas, pois é possível descrever o objeto com mais precisão. Para Alves [39], uma superfície pode ser definida como um elemento matemático que separa o interior do exterior de um objeto. Os objetos tridimensionais gerados pela técnica de modelos de superfície diferem dos modelos por *wireframe* por usarem superfícies tridimensionais, definindo um volume ou contorno de um objeto [36].

Segundo Foggiato *et al*. [36], a modelagem por sólidos é realizada a partir de formas simples bidimensionais como polígonos e círculos ou a combinação destes. Dependendo da geometria desejada, o sólido pode ser gerado fazendo-

se a forma bidimensional percorrer uma trajetória ou ainda pela conexão de duas ou mais destas formas, desenhadas em planos diferentes, sendo admissível também a utilização de comandos baseados em operações booleanas como união, subtração e interseção para a modelagem de sólidos mais complexos. A modelagem sólida não armazena só a geometria do objeto final, mas também todas as formas primitivas e operações usadas para a sua construção. Zeid [38] define um modelo sólido como uma representação completa, única e livre de ambigüidades.

Existem vários métodos de modelagem sólida, sendo que os principais são: CSG (*Constructive Solid Geometry*), B-Rep (*Boundary Representation*), Híbrida, por *Features* e Paramétrica. A maioria dos sistemas de CAD-3D atuais incorporaram em seus núcleos os métodos de modelagem por *features* e paramétrica. As *features* (características) podem ser definidas como elementos das peças que tem algum significado para a engenharia. Segundo Speck [40], a modelagem sólida paramétrica permite a geração de modelos com dimensões vinculadas às variáveis, permitindo a regeneração automática do modelo após cada modificação.

Modelamento Sólido CSG (*Constructive Solid Geometry*): Um modelo CSG é uma árvore binária constituída de objetos primitivos e operadores booleanos. Os primitivos são representados pelas folhas da árvore e os objetos mais complexos são os nós. A raiz da árvore representa o produto completo. Cada primitivo é associado com uma transformação 3D que especifica a posição, orientação e dimensões. Este método caracteriza-se por compor modelos a partir de sólidos [38].

Modelamento Sólido BRep (*Boundary Representation*) [40]: O modelamento BRep é baseado nas técnicas de modelamento de superfícies e são definidas por elementos geométricos primitivos como planos (faces), linhas (arestas) e pontos (vértices) agregados a arcos e superfícies curvas. Assim, ele suporta somente objetos com faces planas e por aproximação, representando faces curvilíneas.

A segunda geração de modeladores BRep incluiu objetos primitivos com superfícies analíticas, como cilindros, esferas, cones, etc., que permitiram a criação de modelos mais elaborados e com geometria "exata". Para tal, foi necessário o uso de algoritmos de intersecção bem mais complexos.

Todos os modeladores geométricos utilizados pelo sistema CAD passaram por uma grande evolução, principalmente os baseados em *features*, como são mostrados na Figura 2.9.

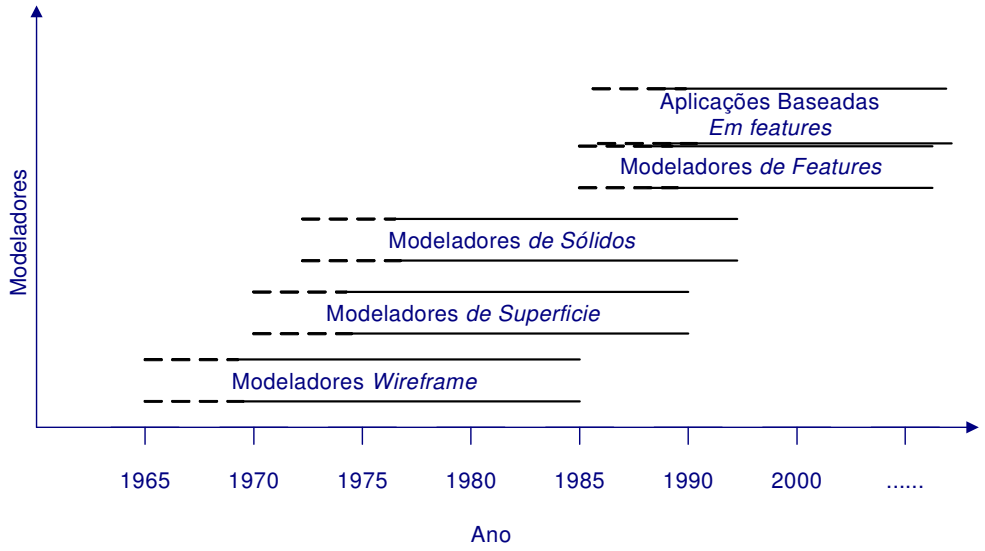


Figura 2.9: Evolução dos modeladores geométricos [41].

2.2.2 Modelamento Sólido baseado em *Features*

Uma *feature* pode ser definida como um elemento físico de uma peça que tem algum significado para a engenharia. Ele deve satisfazer as seguintes condições [42]:

- Ser um constituinte físico de uma peça;
- Ser mapeado para uma forma geométrica genérica;
- Ser tecnicamente significante, sob o ponto de vista da engenharia;
- Ter propriedades previsíveis.

Segundo Schützer [43] *feature* é um objeto, o qual incorpora uma semântica, uma representação paramétrica, conhecimentos tecnológicos e pode ter uma representação geométrica. O significado técnico de *feature* pode envolver a função à qual ela se destina, como pode ser produzida, que ações a sua presença deve iniciar, etc.[44].

As *Features* podem ser pensadas como 'primitivas de engenharia', relevantes a alguma tarefa da área. As *features* possuem uma classificação segundo Eigner *et al* [45] e são encontradas nos sistemas CAD da atualidade (Figura 2.10).

Tipo de Feature	*Descrição*
1. Form feature	Grupo de elementos geométricos. Os recursos salvos com um nome comum podem ser usado se os mesmos elementos geométricos foram aplicadas por diversas vezes. Além disso, é possível ligar recursos de formulário com as propriedades de classes diferentes de propriedade.
2. Semantic feature	Combina as *features* de forma com as semântica que é usada em diferentes fases de desenvolvimento do produto.
3. Implementation feature	Parte das *features* semânticas de uma fase específica do processo de desenvolvimento de produto.
4. Feature for free form surface	- *Features* para o projeto: geometria simplificada, utilizada no início de CAE-fases, contendo as estruturas mecânicas. - *Features* para o projeto de CAD: contêm superfícies livres formados, os recursos poderiam ser usados para incluir restrições aos sistemas contíguos.

Figura 2.10: Classificação das features [45]

O modelamento por *features* vem ganhando terreno, principalmente dentro da engenharia mecânica. O método permite criar furos, chanfros, rasgos, etc., para serem associados com outras entidades ou faces [46].

O modelamento por *features* é baseado na idéia de modelar utilizando blocos de construção. Ao invés de se usar formas analíticas como paralelepípedos, cilindros, esferas e cones como primitivos, o projetista cria o modelo do produto usando primitivos de maior nível, que são mais relevantes para uma aplicação específica. Esta abordagem deveria fazer com que os sistemas de modelamento sólido ficassem mais fáceis de serem usados. Entretanto, o conjunto fixo de *features* oferecido pelos atuais modeladores é muito limitado para uso industrial, o que restringe as possibilidades do projetista. Assim, fica claro que este conjunto deve ser adaptável e extensível aos usuários que trabalham com a biblioteca de *features* [46, 47].

Os esforços para especificação formal de uma linguagem adequada de *features*, iniciados em 1990, proporcionaram que a versão mais nova do STEP incluísse-as com possibilidade de serem definidas pelo utilizador através de uma linguagem padrão de especificação de *features.*

2.2.3 Modelamento Paramétrico

O Modelamento Paramétrico pode ser dividido em dois tipos: O primeiro permite que se criem modelos de produtos com dimensões variáveis. As

dimensões podem ser ligadas através de expressões e variáveis definidas pelo usuário. Como é mostrado na Figura 2.11 [48, 49].

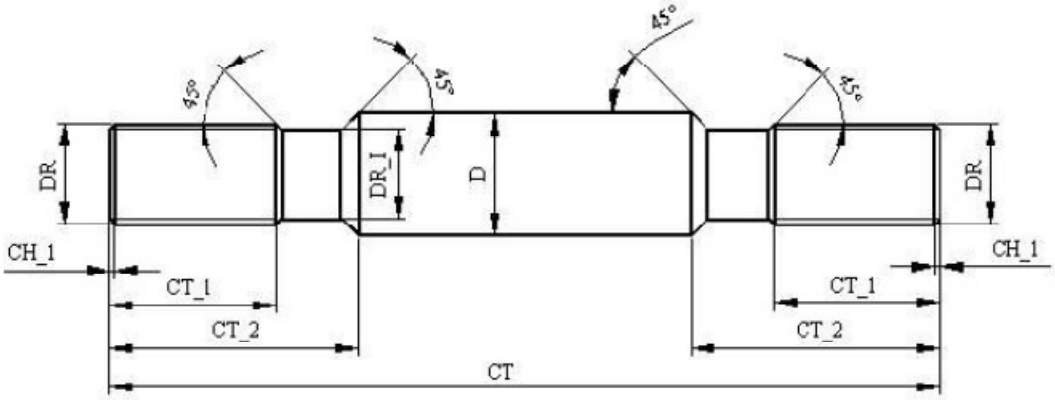


Figura 2.11: Exemplo de modelo paramétrico

O segundo tipo de sistema paramétrico é o que o sistema CAD PRO/Engineer utiliza e que, segundo Ferreira [50], é o único totalmente paramétrico, pois é capaz de gerenciar um determinado modelo utilizado por mais de uma montagem, facilitando assim não só o projeto, mas também as alterações [50].

2.3 Linguagem XML

A linguagem XML (*eXtensible Markup Language*) ou "linguagens de marcação", foi desenvolvida em dezembro de 1997 pelo W3 Consortium, que publicou a versão 1.0 da XML, ela foi criada com a finalidade de descrever dados, assim sendo, seu *framework* permite criar estruturas personalizadas para documentos, agilizando o processo de consulta e exibição das informações. Com a popularização e uso de XML como meio para troca de informações na Web, surgiram os bancos de dados XML, os quais consistem de coleções de documentos XML persistentes que podem ser manipulados.

A aplicação da XML no tratamento de dados bibliográficos é estudada atualmente por diversas empresas e instituições em todo o mundo, com o objetivo de se obterem abordagens mais eficientes para a manipulação de dados na Internet [51].

Segundo Gonzalez *et al*. [52], a XML não é uma linguagem em si, mas uma meta-linguagem: uma linguagem que pode ser usada para criar e descrever outras. A XML tem muitas vantagens sobre outros formatos de dados: é simples, fácil de utilizar, tem licença livre, independe de plataforma, possui um

ótimo suporte (recursos on-line, documentação e softwares disponíveis) e segue as normas internacionais *(*ISO 8879, ISO/IEC 10646, BCP 47, IANA-CHARSETS, etc.*).* A XML tem sido usada com sucesso para criar aplicativos nos diversos domínios da ciência: Matemática (MathML [53]), Química (CML [54]), Geografia (GML [55]), entre outros.

Outro fator que levou a linguagem XML a se tornar um padrão é o fato de ela possuir compatibilidade com mais de um conjunto de caracteres. Este fato torna-se uma vantagem em relação às outras linguagens, que trabalham apenas com padrões americanos, como o ASCII (*American Standard Code* - Código Americano Padrão) [56].

Pode-se afirmar que a estrutura de dados XML é mais poderosa, visto que esta permite a representação de múltiplos elementos do mesmo tipo e de diversos atributos para cada elemento (dados semi-estruturados), facilitando a representação de dados de estrutura mais complexa [57].

A XML tem uma importante característica adicional, uma vez que permite ao autor do documento a definição de suas próprias marcas, possibilitando assim, melhorias significativas em processos de recuperação e disseminação da informação.

A Figura 2.12 mostra uma estrutura básica de um arquivo XML gerado pelo sistema CAE MDESIGN. Podem-se ver os tipos de elementos (vazios, contendo valores ou atributos).

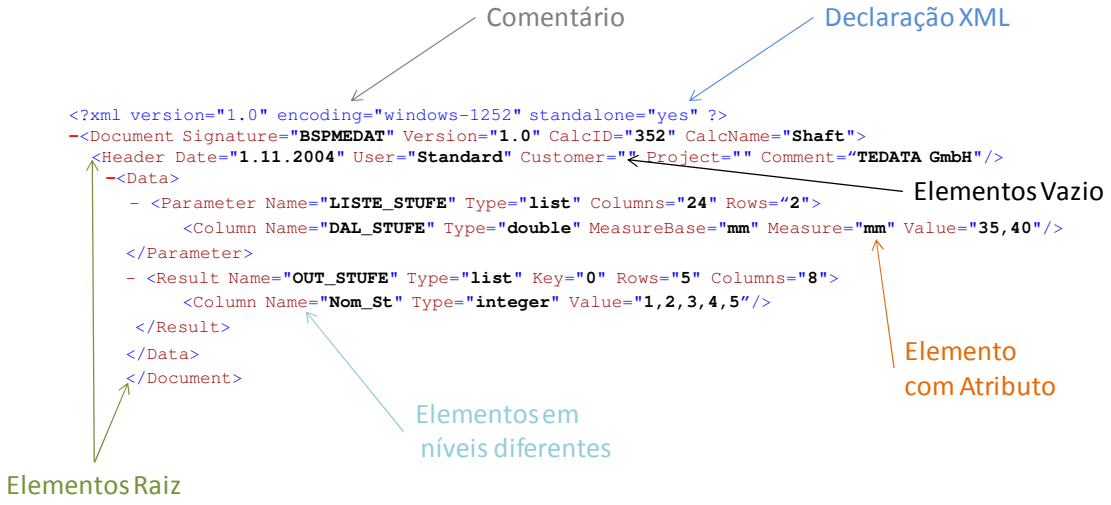


Figura 2.12: Estrutura XML (Elementos e Atributos)

Segundo Kuhl [58] um documento XML contém características especiais que permitem descrever o documento de forma inteligente, tornando o significado de seu conteúdo mais compreensível, tanto para os seres humanos como para os computadores. Basicamente, este documento XML, é composto de três elementos distintos:

- Conteúdo dos dados: são as informações armazenadas entre as *tags* ou elementos;
- Estrutura: a organização dos elementos dentro do documento, que pode possuir diversos tipos de formato, como um memorando ou um contrato, de acordo com as necessidades de marcação da informação;
- Apresentação: é a forma como as informações serão apresentadas ao leitor do documento, podendo ser de diversas maneiras.

Pesquisas recentes mostram que a interface XML oferece uma maneira conveniente e uniforme para integração de dados de engenharia. Além disso, possui flexibilidade e pode ser incorporada em softwares de simulação e visualização de dados de engenharia [59].

Segundo ZHONG e XIAOYAN LI [59] a linguagem XML é muito apropriada para a troca de dados entre diferentes aplicações, sendo este tipo de troca não baseada na premissa de que a definição da estrutura de um grupo de dados está prevista com antecedência.

Com isso, a visualização pode ser extraída através da camada de conversão de dados de engenharia baseado em XML, fornecendo suporte para ambiente de visualização.

A Figura 2.13 mostra o gráfico de relação entre os arquivos transferidos entre diferentes sistemas.

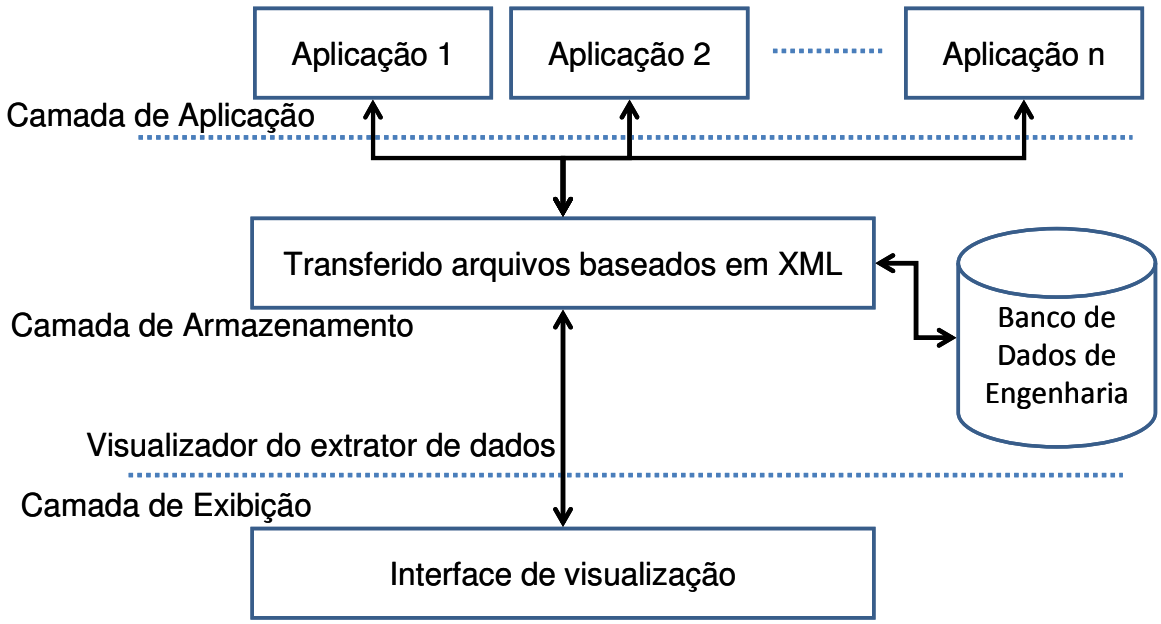


Figura 2.13: Relação de arquivos transferidos entre diferentes sistemas [59]

No desenvolvimento de um módulo para integração dos sistemas CAE/CAD é necessário utilizar um grande número de elementos, o que torna esta tarefa um pouco complicada. Entretanto, os sistemas CAE/CAD atuais de grande porte já trazem algumas ferramentas com o objetivo de tornar essa atividade mais simples [52].

Aplicações padronizadas para XML possibilitam que diferentes aplicativos trabalhem em conjunto, trocando informações entre o documento e proporcionando maior interoperabilidade. Assim, uma empresa que tem um sistema desenvolvido em uma linguagem de programação qualquer (como Delphi, Visual Basic, etc) pode disponibilizar seus dados em um arquivo XML para que outra empresa, tendo um sistema em outra linguagem e instalado num computador com outro sistema operacional, possa acessar esse conteúdo de maneira transparente [58].

Outra utilização encontrada por usuários de todo o mundo para a XML é o armazenamento de informações. Muitos a utilizam desse modo pelo fato desta ferramenta possuir características para estruturação de dados, que serve-se de uma arquitetura que segue o modelo de dados hierárquico. Tais características permitem que dados sejam organizados e trabalhados de forma simples [60].

2.4 Integração CAE/CAD

Em busca de competitividade, as empresas têm procurado adaptar-se às novas tendências do mercado, o que tem exigido também uma grande diversificação dos produtos. Neste contexto, é crucial a diminuição do tempo de lançamento de novos produtos, bem como a redução de custos dos projetos.

A utilização de sistemas CAE (para a concepção) CAD (para o detalhamento) e CAE (para a análise) são fundamentais. Porém, esses sistemas não estão bem integrados, uma vez que as informações conceituais e dos modelos de CAD e CAE são de diferentes tipos, não existindo atualmente um modelo genérico, unificado, que permite tanto conceito como *design* e análise da informação a ser especificada e compartilhada [44].

Vários pesquisadores têm trabalhado em problemas relacionados com a integração dos sistemas CAE com CAD. Kosavinta [61] descreve a integração do CAD com um Sistema de Suporte à Decisão (DSS - *Decision Support System*). O objetivo do DSS é permitir que o projetista possa tomar decisões em tempo real sobre um projeto. O DSS avalia o projeto quanto à viabilidade e custos. Da mesma forma, Shehab [62] detalha a integração de um sistema CAD baseado em conhecimento.

O sistema baseado em conhecimento tem a capacidade de ajudar o projetista a criar um modelo ideal com base no tempo de montagem e custo estimado, bem como fornecer sugestões de melhoria do projeto. Além disso, King [63] utilizou a interface API do sistema CAD SIEMENS NX, Hypermesh e Fluente para criar um CAD-Centric para análise CFD. Estes exemplos de processos de integração CAE com CAD são semelhantes ao proposto neste trabalho, onde os métodos de otimização e a variedade de ferramentas CAE estão sendo integrados com o CAD [64].

Segundo KENWORTHY [64] embora o método de integração realizado nesses exemplos e os programas criados especificamente para integrar os sistemas CAE sejam eficientes para a execução das análises ou aplicativos específicos, tornar-se-ia repetitivo o emprego dessa abordagem para a execução de diversos tipos de aplicações. A solução seria a utilização de uma abordagem mais genérica de integração de processos, que permitam ao projetista interagir com aplicações externas e com o sistema CAD através de um instrumento único personalizado.

Com isso, a integração do sistema CAE com o CAD traz uma grande vantagem em relação ao desenvolvimento de um produto, pois pode-se simular situações reais de seu uso e possibilitar sua otimização, diminuindo o tempo de desenvolvimento, a quantidade de material utilizado para sua produção e até mesmo a simples escolha de um material mais apropriado, aprimorando relações como custo benefício e certificando-se de que sobre determinadas situações tal produto não apresentará problemas. Isso certamente acarretará vantagens para a empresa, tanto no custo e tempo de fabricação, quanto na qualidade [65].

Theis e Trautmann [66] afirmam que aproximadamente 50% do tempo de um engenheiro é gasto procurando por dados que se perdem durante o processo de importação entre os sistemas CAE e CAD. Segundo esses autores, a integração entre o CAD e o CAE possibilitará uma otimização entre a comunicação destes sistemas e trará uma produtividade significativa. No entanto, os usuários de sistemas CAE/CAD têm buscado cada vez mais sistemas que proporcionem facilidades no uso e no aprendizado, optando por comandos intuitivos, auto-explicativos e com o menor número de interações com o mouse e o teclado [66].

Alguns sistemas CAD mais sofisticados - os chamados *"high end"* – já suportam algumas simulações que identificam interferências, colisões, pontos de visualização de perfis, visualização do produto por diversos ângulos e estudo de movimentos, mas não chegam ao nível de sofisticação de dimensionar os elementos que só são possíveis com o CAE. Outros já apresentam os conceitos de associatividade entre os módulos dos sistemas CAD/CAE nativos, ou seja, do mesmo fabricante [67].

2.4.1 Interface de Programação

A maioria dos sistemas CAD permite que seus usuários, através de interfaces de programação, customizem suas aplicações. Para isso, o usuário precisa possuir conhecimentos em programação e desenvolver aplicativos específicos. Esses aplicativos podem ser criados utilizando funções internas do sistema CAD, permitindo a automação de processos.

Algumas vantagens relacionadas à criação de um aplicativo em um sistema CAD são [68]:

- Automatizar processos dentro do sistema;

- Criar janelas de diálogo utilizando a mesma plataforma de interface que o sistema CAD em questão;
- Manipular dados diretamente no modelo nativo, não necessitando de exportações para outros *softwares*;
- Efetuar cálculos através de um novo algoritmo, utilizando parâmetros do modelo nativo;
- Gerenciar os tipos de dados de saída do aplicativo desenvolvido;
- Interface de comunicação com outros programas.

Normalmente, os sistemas CAD de grande porte são dotados de ferramentas que fornecem ao programador um meio de desenvolver a interface do aplicativo com o próprio sistema. A ferramenta mais conhecida para esta função é a API [68].

2.4.2 Interface de Programação de Aplicativos (API)

A Interface de Programação de Aplicativos (API - *Application Programming Interface)* é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por programas aplicativos que não querem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços. De modo geral, a API é composta por uma série de funções acessíveis somente por programação e que permite utilizar características do software menos evidentes ao utilizador tradicional, como por exemplo, sistemas CAD que possuem uma API específica para criar automaticamente entidades de acordo com padrões definidos pelo utilizador [69].

São exemplos de APIs normalmente utilizadas no desenvolvimento de um aplicativo em um sistema CAD [70]:

- API de interface com usuário - Inclui o desenvolvimento de menu, textos de mensagens de apresentação das funções, etc.;
- API para seleção de modelo - Permite uma interação com a interface virtual do modelo geométrico;
- API de manutenção de dados - Acesso aos dados do modelo virtual;

- API de funções - Inclui os cálculos matemáticos, processamento de informação, etc.

Programas com API aberta podem funcionar em dois ambientes diferentes, dependendo de como o programa foi vinculado. Os dois ambientes são [71, 72]:

- Externa: Esses programas são aplicações autônomas que podem ser executados a partir do sistema operacional, fora do sistema CAD. Por esta razão, a API aberta representa uma ferramenta avançada para o desenvolvimento de uma interface que permite controlar os modelos de acordo com uma série de operações desejadas.
- Interna: Esses programas só podem ser executados a partir de uma sessão do sistema CAD. Estes são carregados e incorporados às funções quando o mesmo é processado. Uma vantagem desta abordagem é que os executáveis são muito menores e rápidos [73].

2.4.3 Interface de Programação do sistema CAD Siemens NX

O Sistema Siemens NX possui uma arquitetura de programação aberta (Figura 2.14), ou seja, é possível acessar as funções API existentes dentro do Siemens NX utilizando aplicativos ou programas em diversas linguagens de programação. Assim sendo, o sistema Siemens NX disponibiliza duas ferramentas que auxiliam na criação destes aplicativos: A primeira é a função *User Interface Styler,* (API de interface visual do NX), onde é possível construir a interface gráfica, utilizando o próprio módulo de construção de interface deste sistema CAD. Essa função será detalhada através de um exemplo ainda neste tópico.

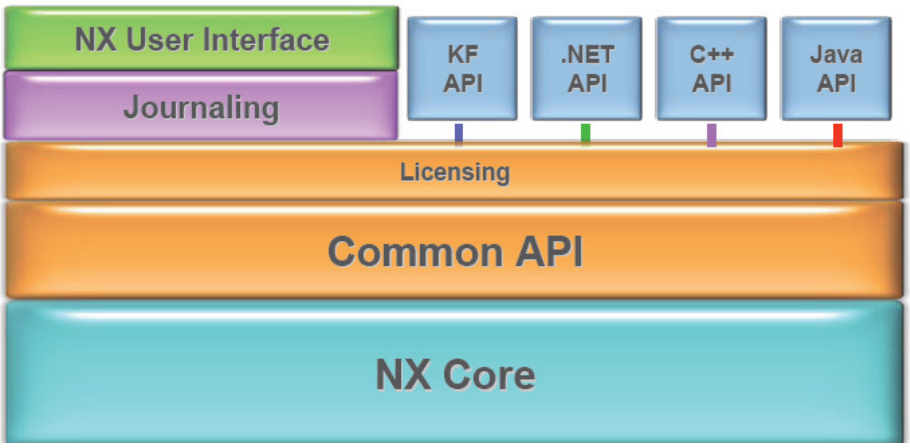


Figura 2.14: Arquitetura aberta do sistema SIEMENS NX [72]

A segunda função é o *journaling*, que é uma ferramenta de automação rápida que grava, edita e reproduz as ações do usuário em uma sessão utilizando o sistema CAD Siemens NX. Com base na linguagem de programação Visual Basic.NET, ele produz um arquivo de script da sessão gravada. Essa sessão pode ser editada e reformulada, utilizando uma programação simples e de fácil utilização. Os arquivos gerados, utilizando essa função, são capazes de serem usados como uma técnica básica para automatizar o fluxo de trabalho repetitivo, uma vez que os programas gerados podem ser usados como um modelo para trabalhar [74].

Para demonstrar o funcionamento de um modelo API do UG NXOpen, será construído uma interface para usuário do sistema CAD Siemens NX. Isso se dá através da função *User Interface Styler.*

Com esta ferramenta, pode-se criar botões e inserir textos informativos em cada ação necessária que o aplicativo requer para o seu funcionamento, entre outros. Após o desenvolvimento de toda a interface visual, esta ferramenta ainda proporciona a construção total de programação que será usada para inserir um programa criado pelo usuário.

A Figura 2.15 apresenta o ambiente de desenvolvimento da interface visual do *software* Siemens NX. A janela de interface é o resultado do processo, ou seja, é o objeto com que o usuário interage quando o aplicativo está em execução.

Com o editor de funções, o programador do aplicativo direciona quais tipos de dispositivos, como botões ou barras de rolagem, além dos textos, serão aplicados à janela de interface. No editor também existe um local onde são inseridos os nomes de chamada de cada dispositivo. Estes nomes são os

endereços utilizados para a inserção de um comando da programação que controla a função desejada. A lista de funções apresenta quais dispositivos serão criados na janela de interface.

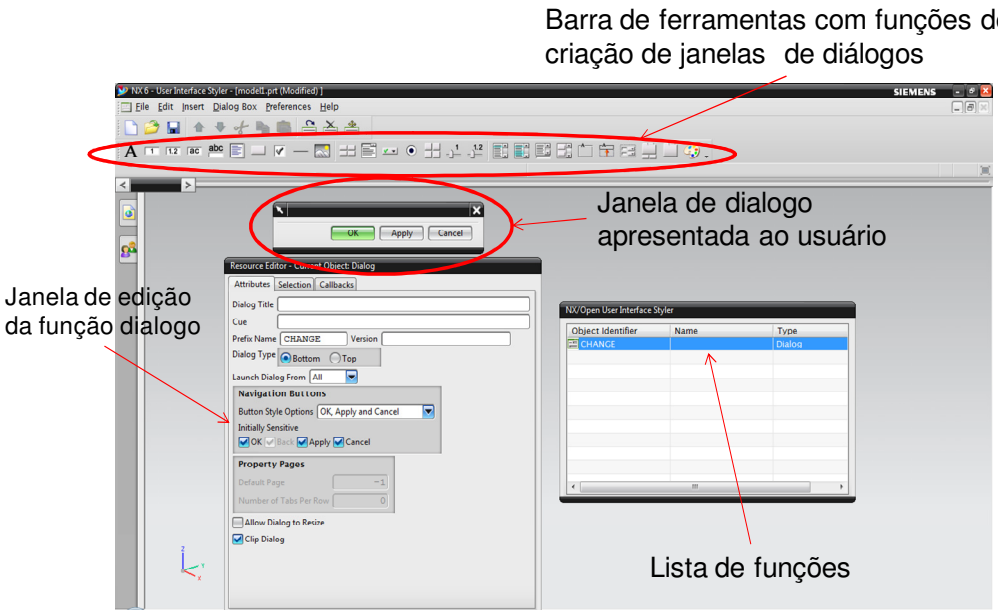


Figura 2.15: Tela de desenvolvimento de interface visual (Siemens NX).

Após o desenvolvimento da janela de interface, é dada ao programador a opção de selecionar qual linguagem de programação será utilizada (Figura 2.16).

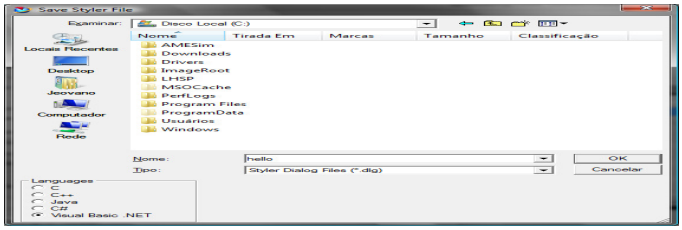


Figura 2.16: Lista de linguagens oferecidas pelo NX.

O *software* NX gera, então, os arquivos que serão utilizados no aplicativo. A Figura 2.17 acima apresenta o resultado desta função para uma janela de interface gerada para um aplicativo que será desenvolvido em linguagem VB.Net. O nome "*hello*" é dado para a criação deste exemplo. O arquivo "*dlg*" é a própria janela de interface e o arquivo "*vb*" é o programa na linguagem VB.net da janela de diálogo.

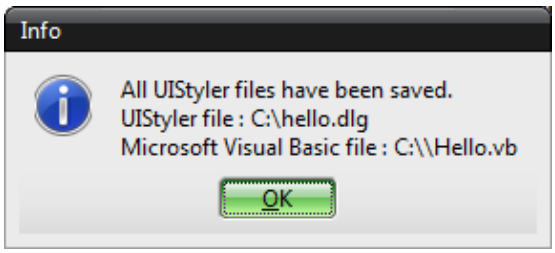


Figura 2.17: Arquivos gerados no desenvolvimento de uma janela de diálogo

Com estes arquivos, pode-se iniciar a utilização de outra ferramenta, como a *Visual Studio*. Esta ferramenta é uma interface de programação que permite aos usuários criar aplicativos customizados utilizando a linguagem de programação VB.Net.

O Microsoft Visual Studio possibilita criar uma interface com o sistema CAD NX. Através deste software é possível construir um projeto onde se insere os arquivos gerados na criação da janela de diálogo obtendo-se uma estrutura básica de programação, conforme a Figura 2.18. O arquivo gerado insere o comando de chamada das funções de manipulação de arquivos XML e de acesso as APIs do NX e do Windows, com a função de início da classe (*Public Class*), cria o vínculo com o software principal. Dentro da "class" são declaradas as variáveis, funções de criação e as linhas de comandos.

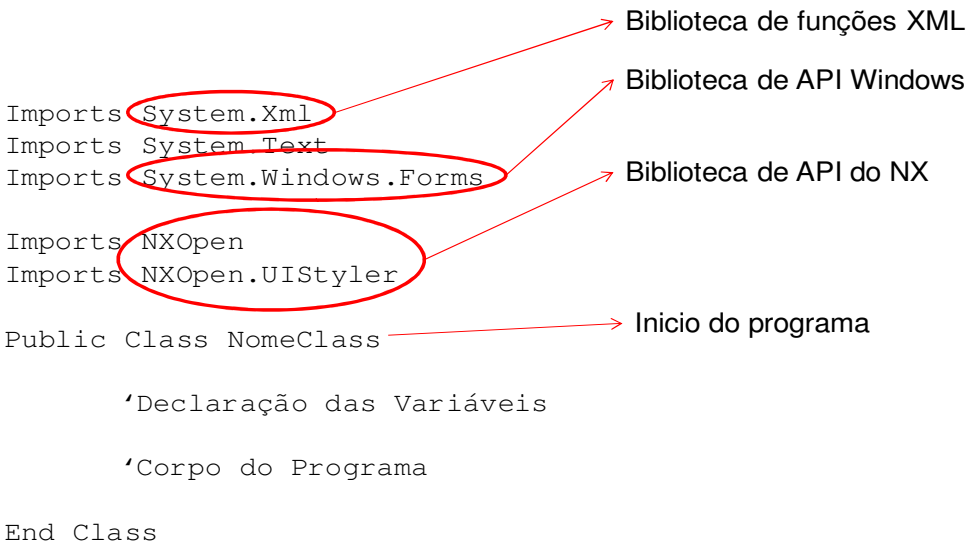


Figura 2.18: Exemplo de estrutura de programação de um janela de diálogo.

3 Proposta de Trabalho

Neste capítulo são apresentados o objetivo e o método de trabalho.

3.1 Objetivos

Este trabalho tem como objetivo desenvolver um aplicativo para integração entre uma ferramenta de dimensionamento de componentes mecânicos e um sistema CAD, visando à geração automática do modelo 3D, a partir dos resultados do dimensionamento (XML), a diminuição de tempo e a eliminação dos erros na manipulação dos dados. Os objetivos específicos são:

- Aprofundamento dos conceitos relativos à utilização de arquivos XML para integração entre sistemas CAD/CAE;
- Aprofundamento na interface de programação do sistema CAD Siemens NX;
- Implementar o leitor de arquivos XML no sistema CAD e uma interface de integração CAE/CAD;
- Avaliar a interface do aplicativo para verificar os ganhos na utilização do mesmo.

Não faz parte dos objetivos avaliar a exatidão dos dados gerados pelo sistema CAE, nem a validação do aplicativo como um software.

3.2 Método de Trabalho

Para atingir tais objetivos foram feitas pesquisas bibliográficas, sendo elas sobre a integração entre sistemas CAD e CAE e a utilização de arquivos XML para integração de informações entre sistemas CAx.

Foram gerados e executados pequenos aplicativos como treinamento na interface de programação do sistema CAD NX e linguagem de programação VB.Net.

Para a integração proposta, inicialmente foi implementado uma rotina para leitura dos arquivos XML, que posteriormente foi utilizada para criação de uma interface que possibilitou a comunicação entre os dois sistemas, utilizando a linguagem XML como ligação.

Foram realizadas atividades práticas utilizando o sistema CAE MDESIGN e o sistema CAD Siemens NX, que possui uma interface de programação aberta,

sendo possível criar aplicativos utilizando a linguagem de programação VB.Net dentro deste sistema.

Para a avaliação do aplicativo e da interface foi utilizado o módulo de cálculo de eixos existente no sistema MDESIGN, que é um dos elementos mais comuns em projetos mecânicos, sendo que o mesmo conceito utilizado para este módulo pode ser aplicado para qualquer um dos módulos contidos no sistema MDESIGN.

A avaliação foi feita por três alunos, sendo estes de mestrado e graduação, com conhecimento no sistema Siemens NX, utilizando como base o relatório de três eixos com diferentes níveis de dificuldade, gerado pelo sistema MDESIGN, para o modelamento geométrico no sistema CAD Siemens NX como normalmente é feito nas empresas que utilizam estes sistemas.

Os métodos foram realizados seguindo uma seqüência de etapas como é mostrado no diagrama (Figura 3.1).

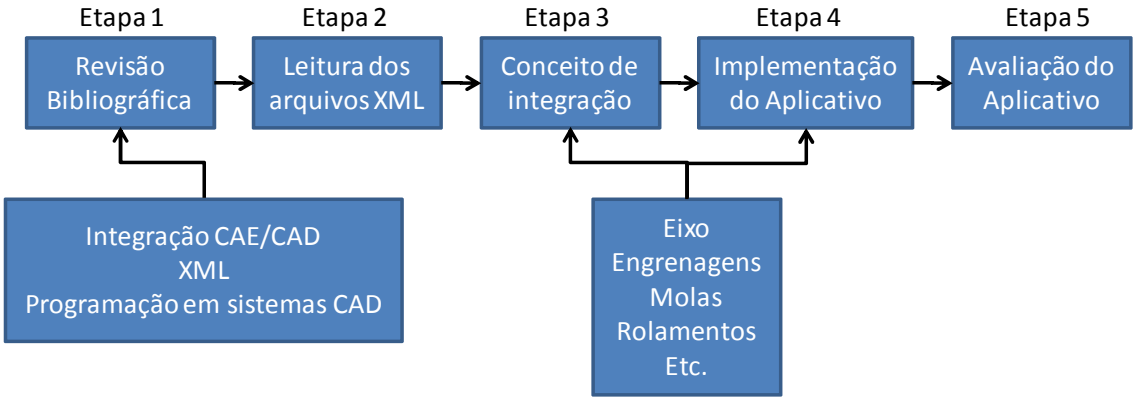


Figura 3.1: Diagrama de etapas

4 Desenvolvimento e Implementação do Aplicativo

Neste capitulo, é apresentado o desenvolvimento do conceito, implementação e análise dos resultados, conforme objetivos apresentados no capitulo 3.

4.1 Integração CAE e CAD – Conceito

Os cálculos de elementos de máquinas utilizando métodos analíticos são tarefas básicas de um engenheiro. Para isso já existem ferramentas computacionais que o auxiliam neste tipo de trabalho, que vão desde uma simples planilha eletrônica até softwares dedicados, porém, estas ferramentas normalmente limitam-se ao desenvolvimento de cálculos, não se integrando com os sistemas CAD, sendo que o projetista necessita utilizar as informações obtidas através dos cálculos para construir o modelo 3D.

A obtenção dos dados geométricos é feita de forma manual, ou seja, o projetista identifica essas informações utilizando os arquivos de respostas do sistema CAE, por exemplo, ao dimensionar um elemento de máquina utilizando um sistema CAE qualquer, o projetista obtém um relatório com os dados dimensionados, na forma virtual (arquivos em PDF) ou física (Impressa), com esses resultados ele identifica os dados geométricos do elemento dimensionado. Então com essas informações ele pode iniciar um novo modelo ou alterar um modelo já existente no sistema CAD (Figura 4.1).

A Figura 4.1 mostra o fluxo das informações entre os sistemas CAE e CAD utilizando o método manual.

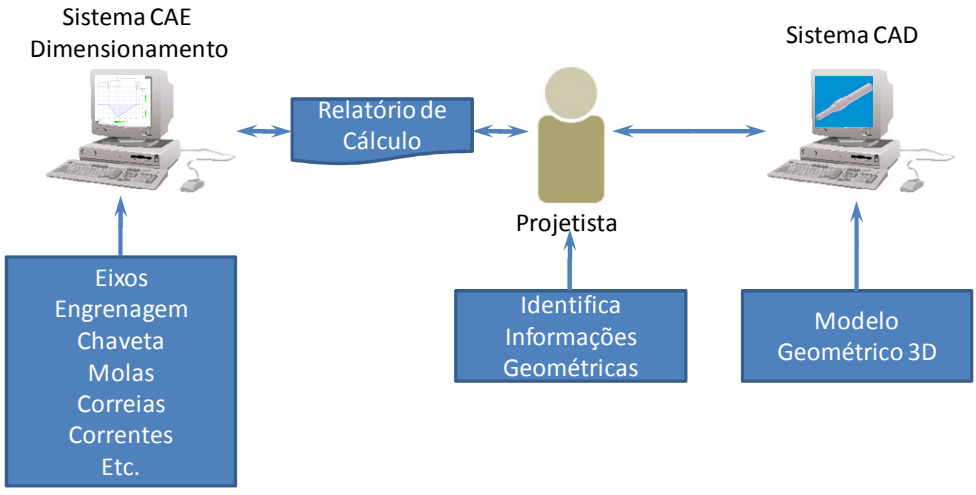


Figura 4.1: Fluxo de informação no processo manual entre os sistemas CAE e CAD

Como já foi citado no capítulo 2, esse processo de dimensionamento e geração do modelo geométrico feito sem uma integração dos sistemas CAE e CAD trazem as seguintes desvantagens; A transferência de dados por ser feito manualmente pelo usuário aumenta significativamente as chances de ocorrerem erros nessa transferência, bem como a sua demora no processo e a falta de um padrão do modelamento.

Os erros mais comuns que podem ocorrer durante a transferência de dados são os de má interpretação do relatório, com a utilização de informações erradas, ou como em alguns casos, existem dimensões que o sistema CAE calcula implicitamente e não exibe no relatório de saída, sendo necessário que o projetista repita esses cálculos manualmente para obter estas dimensões.

O aumento do tempo nessas fases do projeto é maior, pois e feita de forma manual, se houver erros há a necessidade de refazer e depende da experiência do projetista.

A falta de padrão no modelamento ocorre, devido ao ser humano não executar uma tarefa da mesma forma que outro ser humano, assim um projetista não modela da mesma forma que outro projetista, causando uma falta de padrão na geração dos modelos geométricos, influenciando no tempo de execução do projeto.

Com todas estas desvantagens identificadas este trabalho teve como objetivo trazer uma colaboração para a integração entre sistemas CAE analíticos e sistemas CAD.

Para a integração foi analisado o sistema CAE e CAD de forma individual, e depois de entendido os dois sistemas foi desenvolvido um conceito para essa integração.

Para os Sistemas CAE analíticos, foram identificadas as formas de saídas de dados que o mesmo gera para os dimensionamentos de elementos mecânicos, com isso a maneira encontrada como padrão para obtenção dos dados geométricos foi o formato de arquivo XML.

Todos os sistemas CAE comerciais de dimensionamento analítico possuem esse tipo de extensão como saída/resultado do dimensionamento. Este tipo de arquivo possui uma estrutura que possibilita fácil acesso as informações necessárias para geração das geometrias como já apresentado no capitulo 2.

Já para os sistemas CAD foram verificadas as interfaces de programação, todos os sistemas CAD de médio e grande porte oferecem essa possibilidade. A única maneira de integrar o sistema CAE com o CAD e desenvolver um aplicativo, que lê os dados de um arquivo de resposta do sistema CAE e gera o modelo 3D no sistema CAD automaticamente, eliminando o processo realizado pelo projetista, esse fluxo e mostrado na Figura 4.2.

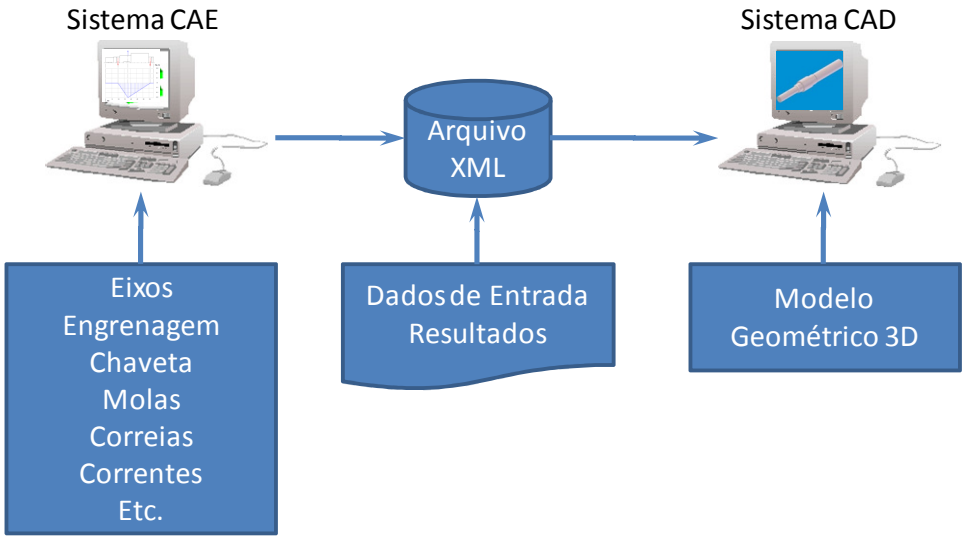


Figura 4.2: fluxo de dados utilizando arquivo XML como integrador

O conceito desenvolvido neste trabalho pode ser aplicado para qualquer sistema CAE Analítico que tenha um arquivo de dados de resposta no formato XML. Esse mesmo conceito pode ser extrapolado para outros sistemas CAx podendo ser criado um ambiente único (Figura 4.3)

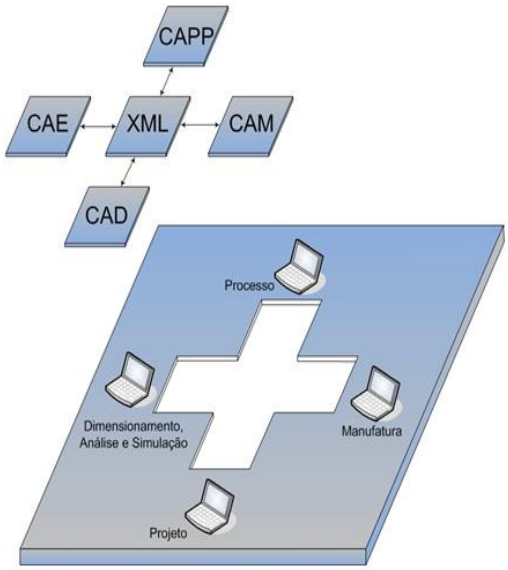


Figura 4.3: Integração dos sistemas CAx utilizando XML

No tópico 4.2 é apresentado o desenvolvimento de um aplicativo para geração de eixos utilizando o conceito de integração entre os sistemas CAE e CAD, esse aplicativo é um exemplo podendo ser aplicado para os outros módulos do sistema CAE utilizado.

4.2 Integração CAE e CAD – Conceitos Aplicados ao Sistema CAE MDESIGN e Sistema CAD SIEMENS NX

Com base no conceito de integração apresentado no tópico anterior o presente trabalho apresenta o desenvolvimento de um aplicativo para integração entre os sistemas CAE e CAD, utilizando os softwares MDESIGN e Siemens NX. Esta integração consiste em uma interface que possibilita a comunicação entre os dois softwares utilizando a linguagem XML como ligação, permitindo que a partir de um cálculo realizado no software CAE MDESIGN seja possível a geração automática do modelo geométrico 3D no software Siemens NX e futuramente com os outros sistemas computacionais.

O sistema CAE MDESIGN é largamente utilizado em empresas de projetos mecânicos, pois possui módulos de cálculo para todos os elementos padrões utilizados neste tipo de projeto.

O conceito da integração entre os sistemas CAE/CAD aplicado neste trabalho é apresentado na Figura 4.4, onde se pode observar a seqüência do processo. O sistema CAE é responsável pelo dimensionamento dos elementos de máquina que por sua vez gera o arquivo XML (dados de entrada + resultados de cálculo), contendo as informações do cálculo (Módulo de cálculo, material, geometria, carregamentos, resultados, etc.). A estrutura desse arquivo será detalhada no tópico 4.2.1.

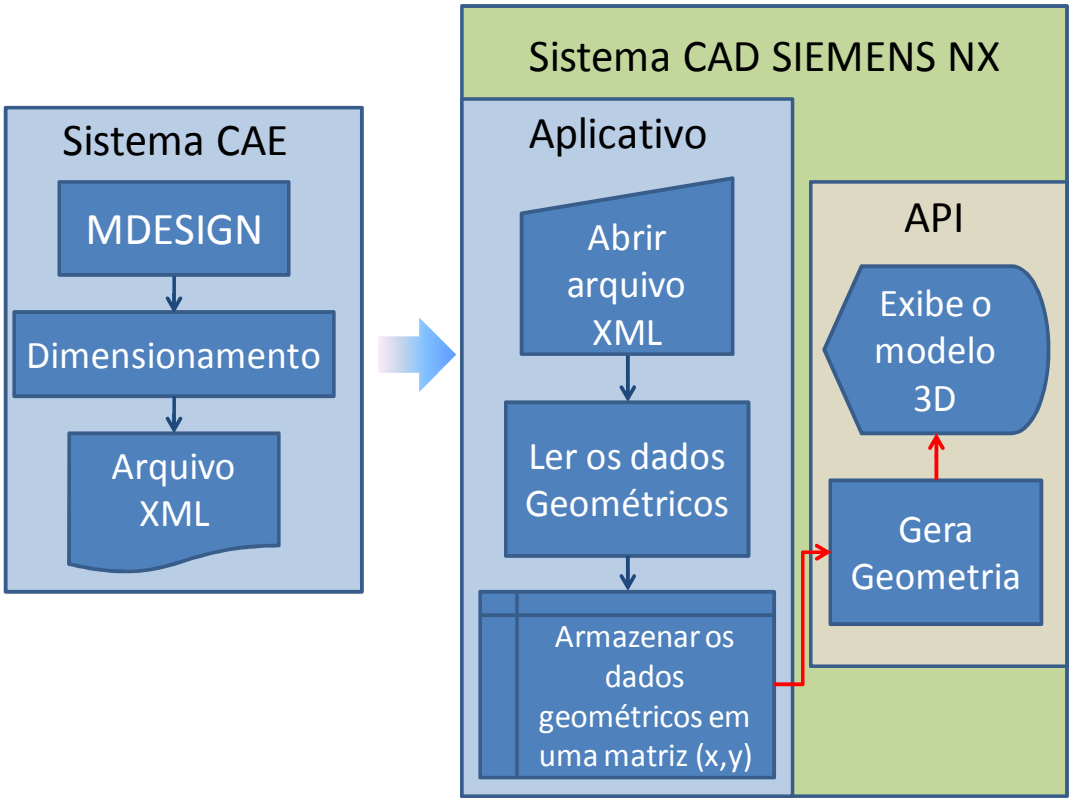


Figura 4.4: Conceito aplicado na integração

O aplicativo é do tipo interno, ou seja, funciona somente com o sistema CAD Siemens NX. Basicamente, ele tem a função de abrir o arquivo XML gerado pelo sistema CAE, ler e extrair as informações necessárias do arquivo XML – neste caso informações geométricas – e armazenar essas informações em uma matriz – que será apresentada no tópico 4.2.2. O código fonte do aplicativo poderá ser visto no Anexo 2.

Uma vez armazenados os dados geométricos, o aplicativo acessa as funções de geração de modelo 3D na API do sistema CAD Siemens NX e, de acordo com o elemento de máquina (eixo, engrenagens, polias, etc.), o modelo geométrico 3D referente aos dados do arquivo XML é gerado e apresentado ao usuário.

O presente trabalho utilizou o módulo de eixo para representação do conceito de integração, porém o mesmo conceito pode ser aplicado a qualquer módulo existente no sistema MDESIGN. Com isso os tópicos seguintes estarão focados no módulo de eixos.

4.2.1 Leitura e armazenamento dos dados do arquivo XML

Para que o aplicativo possa desempenhar as funções que foram designadas, primeiramente foi necessária a implementação de um algoritmo que possibilitou a leitura dos dados do arquivo XML gerado pelo sistema CAE MDESIGN. Estes arquivos são selecionados pelo usuário.

Para a leitura dos dados do arquivo XML, foi utilizada a biblioteca de funções "*System.Xml*" e a biblioteca de API do Windows "*System.Windows.Forms*". Estas bibliotecas são responsáveis por comandos específicos para manipulação de arquivos XML e API.

A Figura 4.5 mostra os comandos utilizados para identificar, abrir e ler o arquivo XML. Os comandos que estão destacados em preto e vermelho são filtros para iniciar a busca pelo arquivo no *drive* d:/ e delimitar a busca por arquivos do tipo XML.

```
openFileDialog1.InitialDirectory = "d:/"
openFileDialog1.Filter = "xml files(*.xml)|*.xml| All Files(*.*)|*.*"
openFileDialog1.FilterIndex = 2
openFileDialog1.RestoreDirectory = True
If openFileDialog1.ShowDialog() = System.Windows.Forms.DialogResult.OK
Then
theUI.NXMessageBox.Show("Arquivo",
NXMessageBox.DialogType.Information, openFileDialog1.FileName)
reader = New XmlTextReader(openFileDialog1.FileName)
```

Figura 4.5: Linhas de comando para leitura do arquivo XML

Os comandos que estão destacados em azul partem de uma tomada de decisão, que verifica se o arquivo foi lido. Se isso for verdade, o comando seguinte exibe uma mensagem utilizando uma API do *Windows form*, contendo o caminho do arquivo XML lido, conforme Figura 4.6.

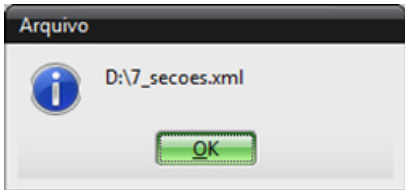


Figura 4.6: Janela de mensagem contendo o caminho do arquivo XML lido

E o comando destacado em verde é o comando responsável por salvar o arquivo XML em uma variável local, que será utilizada posteriormente para identificar as variáveis geométricas que o arquivo contém.

Os arquivos gerados pelo sistema CAE MDESIGN possuem uma estrutura (conforme a Figura 4.7), um exemplo pode ser visto no Anexo 1. Os elementos encontrados nos arquivos XML são divididos em dois grupos: Parâmetros de Entrada e Resultados.

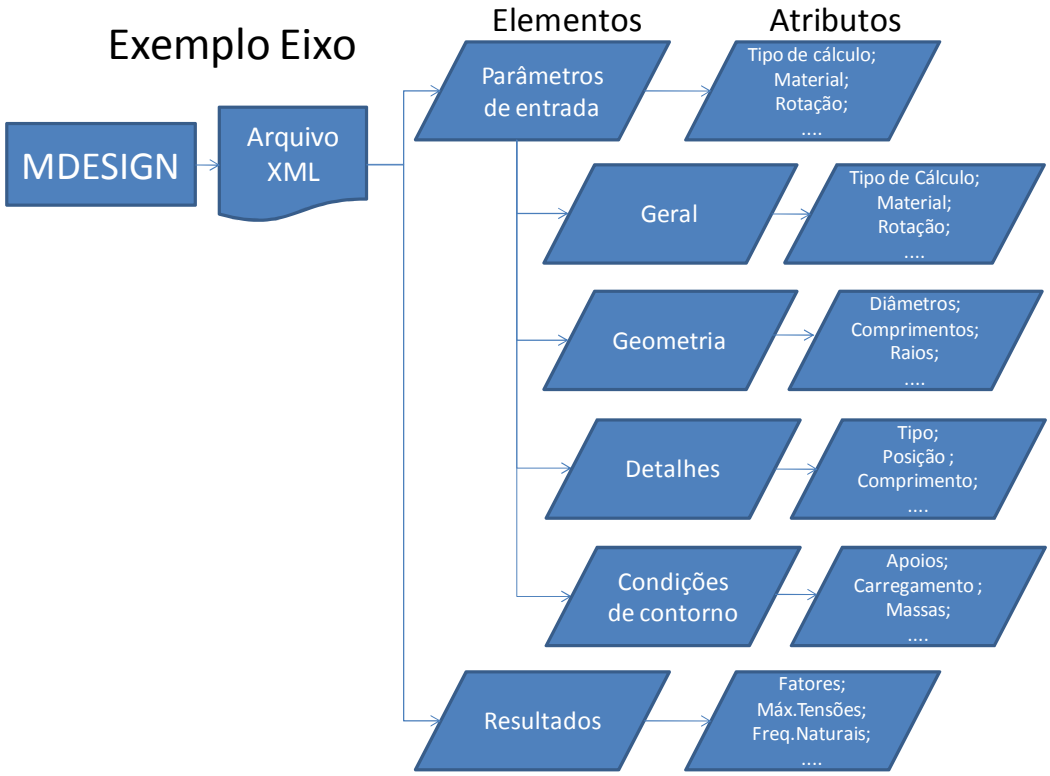


Figura 4.7: Exemplo da estrutura XML gerada pelo MDESIGN

O grupo parâmetros de entrada se divide em quatro subgrupos: Geral, Geometria, Detalhes e Condições de Contorno. Todos os subgrupos contêm variáveis utilizadas no dimensionamento e são informadas pelo usuário. O grupo geral contém informações como módulo de cálculo (eixo, engrenagens, polias, molas, etc.), tipo do cálculo (dinâmico, estático, etc.), material (aço, alumínio, polímeros, etc.), entre outras informações, que diferem dependendo do módulo selecionado.

O subgrupo geometria contém apenas informações geométricas (diâmetro, comprimento, altura, raios, etc.), que foram utilizadas neste trabalho. Este é complementado pelo subgrupo detalhes, que basicamente contém informações

como forma do detalhe (Chaveta, Rebaixo, Entalhado, etc.) e geométricas dos detalhes.

Já o subgrupo condições de contorno contém informações sobre os carregamentos (forças axiais e radiais, localização dos mancais, massas, momentos, etc.). E por fim, o grupo resultado contém todas as informações resultantes do cálculo (esforços máximos e mínimos, fator de segurança, etc.).

4.2.2 Lógica de funcionamento do aplicativo

Uma vez feita à leitura do arquivo XML, o mesmo precisa ser delimitado, pois para esse trabalho os elementos e atributos necessários são os geométricos e detalhes (Figura 4.7), independentemente do tipo de elemento de máquina que foi dimensionado.

4.2.2.1 Identificação do módulo de cálculo

Para identificação do módulo de cálculo foi utilizado o seguinte algoritmo (conforme diagrama de atividades da Figura 4.8) onde, após ler o arquivo XML, faz-se uma varredura dos elementos para identificar a qual módulo pertence este arquivo.

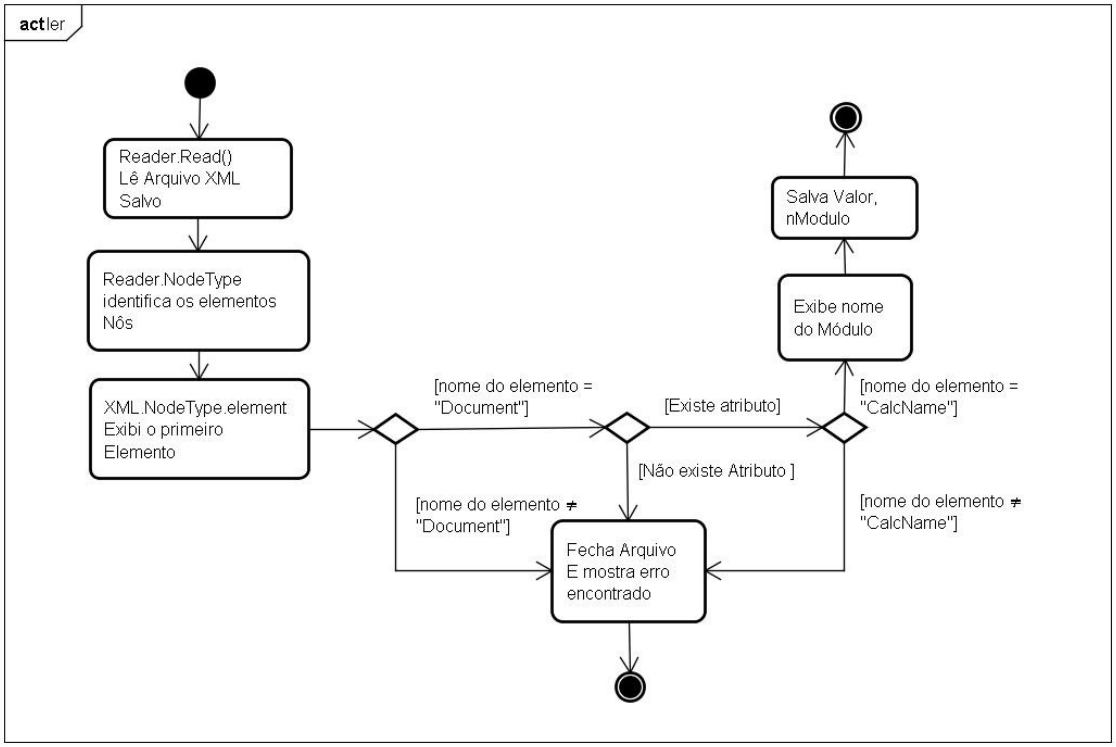


Figura 4.8: Fluxograma de identificação do módulo MDESIGN

As funções descritas no algoritmo de identificação do módulo são:

- *Reader.read()* – responsável pela leitura do arquivo XML;
- *Reader.nodetype* – retorna o nome e valor dos nós;
- *XML.nodetype.elemento – retorna o nome e valor do elemento.*

Com base nessas funções, o aplicativo primeiro lê o arquivo, depois identifica os nós e em seguida os elementos. Uma vez encontrados os elementos, utiliza-se uma estrutura condicional do tipo *if then* para identificar o elemento "*Document*", responsável pelas informações do módulo. É feita outra verificação para descobrir se este está vazio ou contém atributos e, caso haja, um novo elemento é procurado, pois a verificação dos elementos está sendo feita em outro nível hierárquico dentro de *"Document" (*conforme Figura 4.9), se não houver elementos ou atributos em qualquer das pesquisas, uma mensagem de erro é mostrada ao usuário.

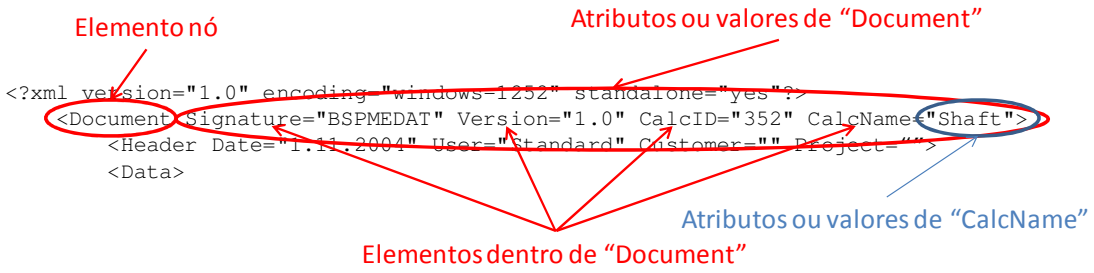


Figura 4.9: Descrição dos elementos e atributos de identificação do módulo

Seguindo o algoritmo, novamente uma estrutura condicional do tipo *if then* é utilizada para identificar o elemento *"CalcName"*, responsável por conter o nome do módulo. Ao encontrá-lo, é exibida uma mensagem com essa informação (Figura 4.10).

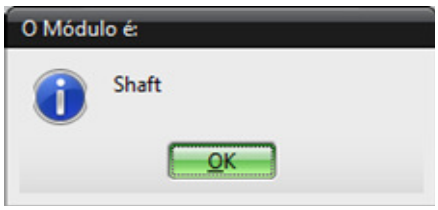


Figura 4.10: Mensagem contendo o nome do módulo (shaft).

4.2.2.2 Identificação das informações geométricas

Assim que o módulo é identificado, os algoritmos da Figura 4.11 e Figura 4.12 são utilizados, os quais são responsáveis pela separação das informações geométricas das seções e dos detalhes. Este algoritmo possui um filtro que identifica somente as informações pertinentes à geometria. Estas informações pertencem a um grupo de elementos específicos associado ao módulo de cálculo, que neste caso foi adotado como exemplo o módulo de dimensionamento de eixos.

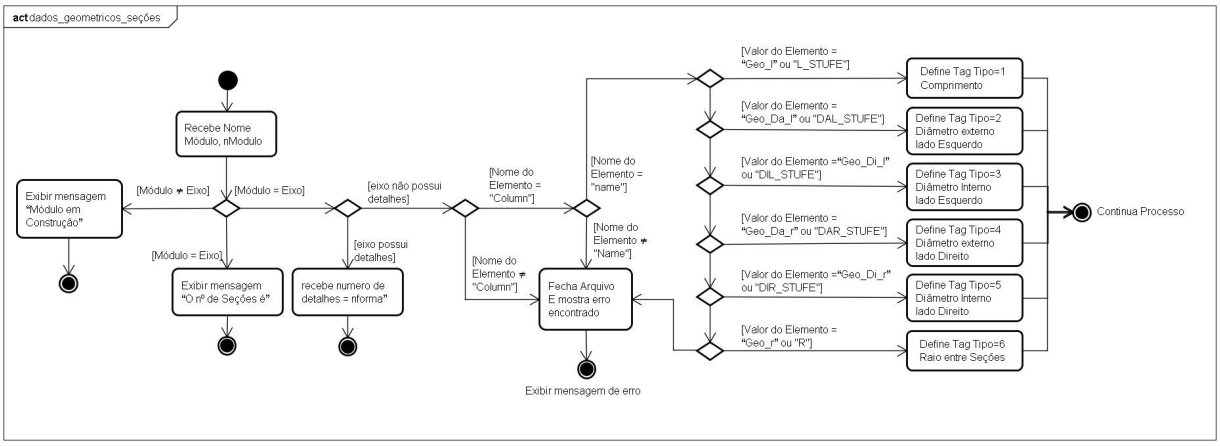


Figura 4.11: Refinamento dos dados geométricos para tags geométricas

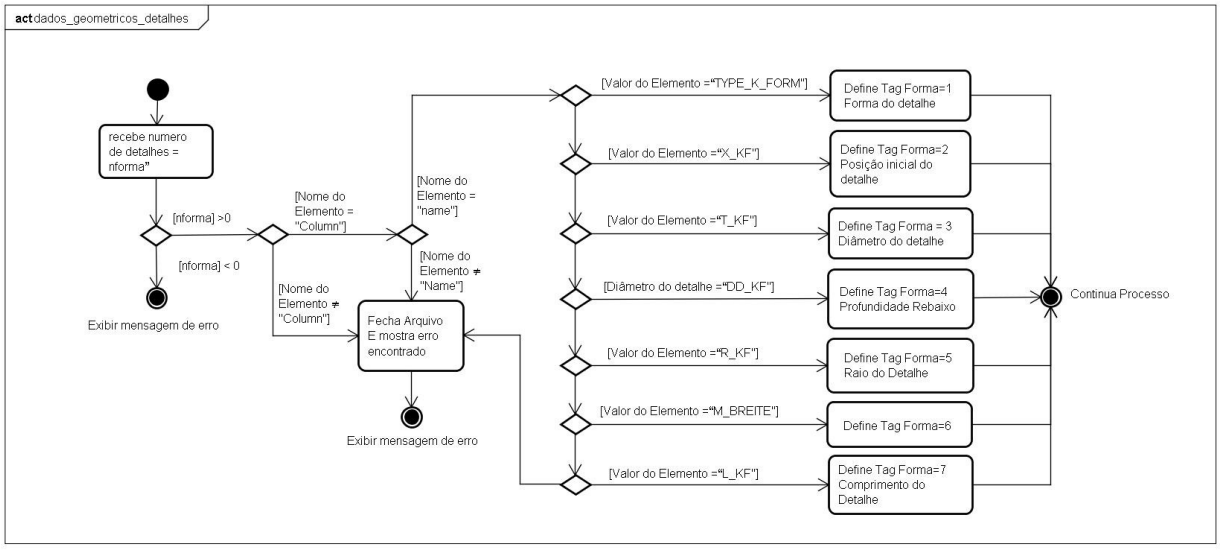


Figura 4.12: Refinamento dos dados geométricos para tags detalhes.

Para o módulo de eixos o grupo de elementos geométricos básicos é identificado pelos elementos (Colunm Name = "GEO_" ou Colunm Name =

"_STUFFE"). Já o grupo de elementos geométricos dos detalhes é identificado pelos elementos (Colunm Name = _KF), conforme Figura 4.13.

Figura 4.13: Descrição dos elementos e atributos geométricos básicos e dos detalhes.

Com base nos elementos geométricos, o algoritmo descrito na Figura 4.11, utilizando estruturas condicionais, identifica o elemento *"Column"* e em seguida o elemento *"Name"*, para então identificar seus valores ou atributos. Esse algoritmo filtra somente os dados geométricos básicos para construção de eixos, que são os diâmetros, comprimento e raio de cada seção, já o algoritmo da Figura 4.12 tem a função de filtrar somente os dados geométricos dos detalhes (forma do detalhe, posição, etc.).

As dimensões básicas e os detalhes são classificados conforme a Figura 4.14, de modo a receberem uma *tag* utilizada para criação de uma matriz geométrica e uma matriz dos detalhes.

Para o dimensionamento básico são utilizados 6 parâmetros que são alocados em *tags* que vão de 1 a 6, estes podem ser encontrados nas versões Inglês e Alemão do sistema MDESIGN (Figura 4.14). Já o dimensionamento com os detalhes, que são (chavetas, rebaixos, entalhes, etc.) agregados ao básico, possuem 7 parâmetros que são alocados em *tags* que vão de 1 a 7, estes podem ser encontrados somente na versão em Alemão do sistema MDESIGN (Figura 4.14).

Tags Geométricas das seções do Eixo	
Tipo (Inglês ou Alemão)	***Tag***
Comprimento (Geo_l ou L_STUFE)	**1**
Diâmetro Externo lado Esquerdo (Geo_Da_l ou DAL_STUFE)	**2**
Diâmetro Interno lado Esquerdo (Geo_Di_l ou DIL_STUFE)	**3**
Diâmetro Externo lado Direito (Geo_Da_r ou DAR_STUFE)	**4**
Diâmetro Interno lado Direito (Geo_Di_r ou DIR_STUFE)	**5**
Raio entre as Seções (Geo_r ou R)	**6**
Tags Geométricas dos detalhes do Eixo	
Tipo (Alemão)	*Tag*
Forma do detalhe (TYPE_K_FORM)	**1**
Posição inicial do detalhe (X_KF)	**2**
Diâmetro Interno (DD_KF)	**3**
Profundidade do Rebaixo (T_KF)	**4**
Raio do detalhe (R_KF)	**5**
Largura do Rebaixo(M_BREITE)	**6**
Comprimento do detalhe (L_KF)	**7**

Figura 4.14: Parâmetros da matriz geométrica para seções e detalhes

Após definir as *tags,* o diagrama de atividades da Figura 4.16, que é uma continuação do diagrama da Figura 4.11, identifica o elemento *"Name"*. Em seguida, o algoritmo, pelo comando `arryRows = Split(reader.Value, ",")` lê os valores ou atributos que estão entre as vírgulas e transforma-os em um vetor. Assim, utiliza uma estrutura de repetição (*for loop),* para fazer o preenchimento da matriz geometria (tipo, i), conforme (Figura 4.15).

```
If (reader.Name = "Value") Then
        'Cria Matrix geometria
        Dim arryRows() As String
        arryRows = Split(reader.Value, ",")
        For i = 0 To UBound(arryRows)
                geometria(tipo, i) = arryRows(i)
                numcol = i + 1
        Next
End If
```

Figura 4.15: estrutura de repetição (for loop) para preenchimento da matriz

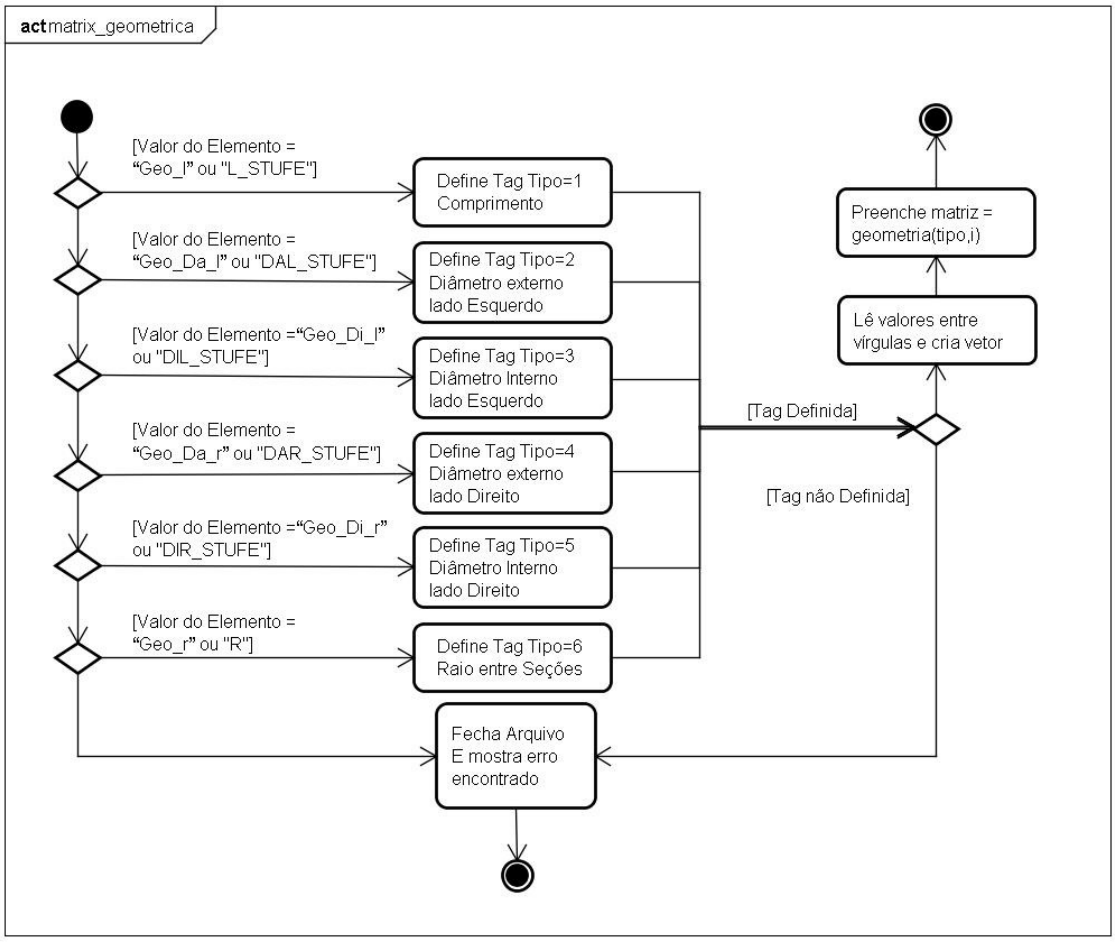


Figura 4.16: Diagrama de atividades da geração da matriz geométrica.

O diagrama de atividades da Figura 4.17 que é uma continuação do diagrama da Figura 4.11, identifica o elemento *"Name"*. Em seguida, o algoritmo, pelo comando `arryRows = Split(reader.Value, ",")` lê os valores ou atributos que estão entre as vírgulas e transforma-os em um vetor. Também utiliza uma estrutura de repetição (*for loop),* para fazer o preenchimento da matriz detalhe (tipo, i), sendo seu código fonte semelhante ao mostrado na (Figura 4.15).

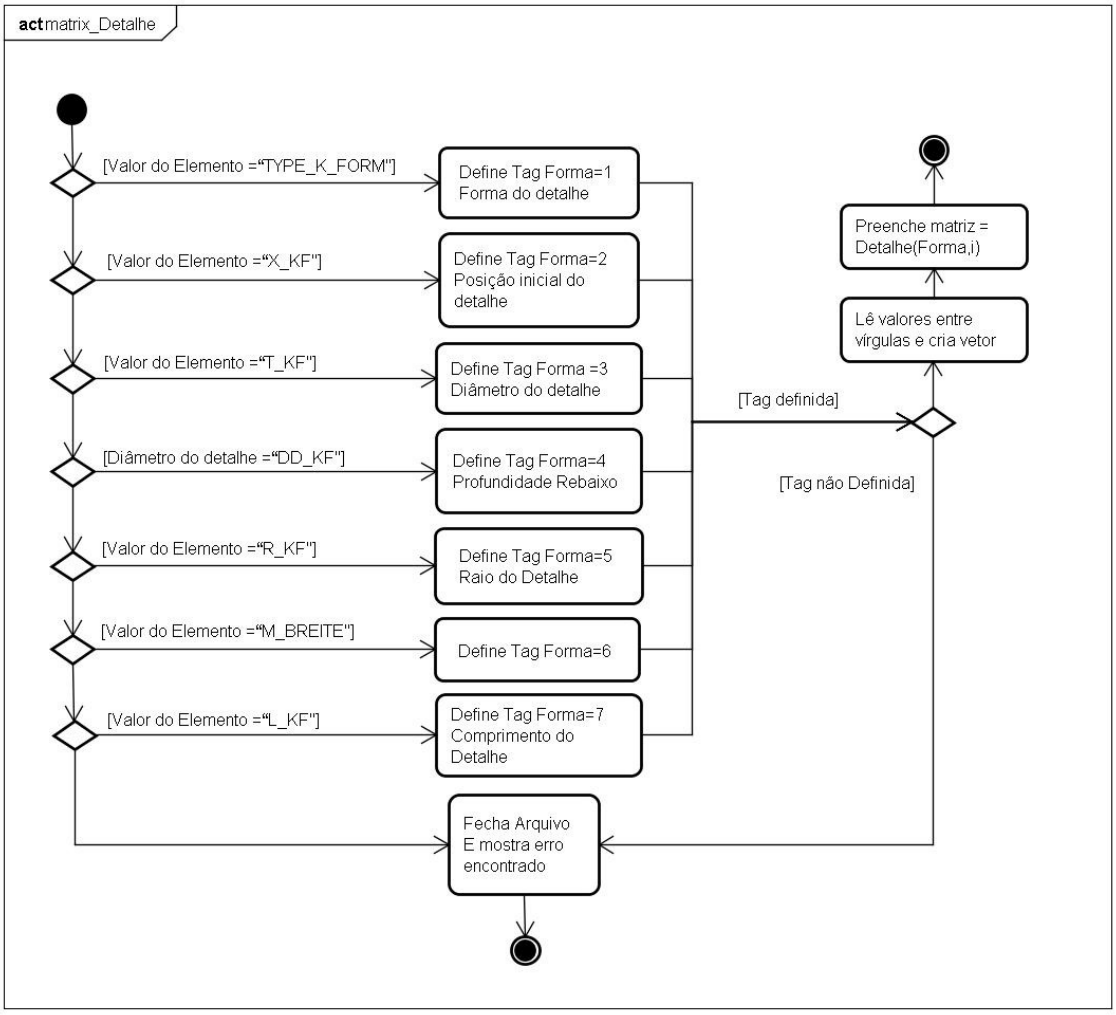


Figura 4.17: Diagrama de atividades da geração da matriz detalhe.

A matriz geométrica pode ser representada conforme Figura 4.18 onde o número máximo de seções é 50, devido à limitação do sistema CAE MDESIGN. As colunas da matriz representam as seções do eixo e as linhas os parâmetros geométricos.

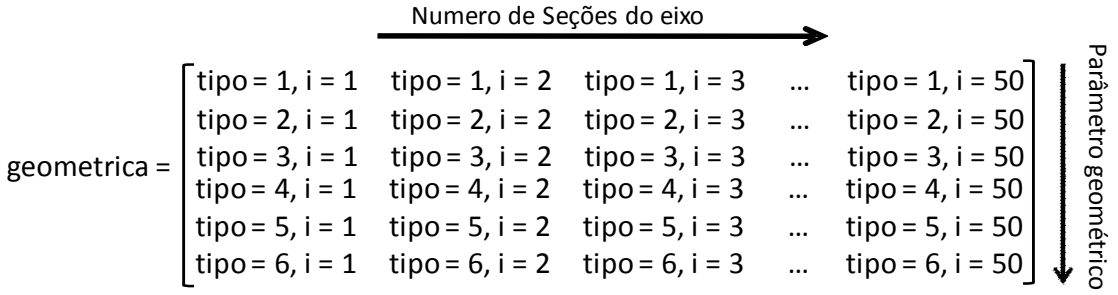


Figura 4.18: Matriz geométrica

A matriz detalhe é semelhante à apresentada na Figura 4.18, porem possui outras limitações onde o número máximo de detalhes é 30, devido à limitação do sistema CAE MDESIGN, a matriz detalhe e apresentada na Figura 4.19, onde as colunas da matriz representam os detalhes do eixo e as linhas os parâmetros geométricos dos detalhes.

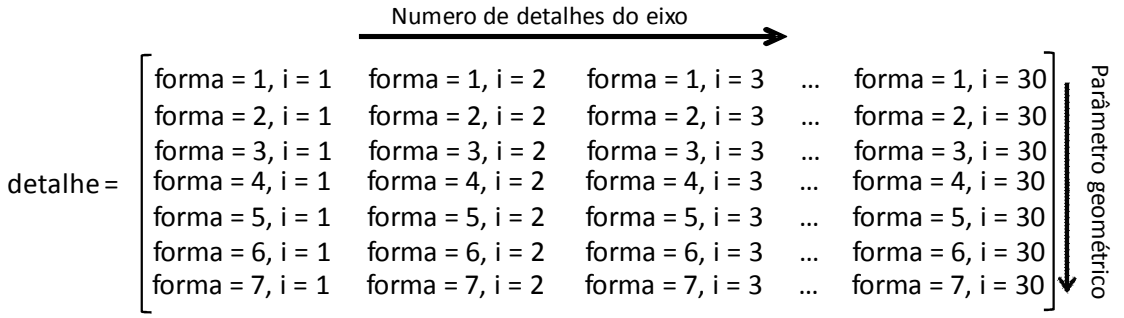


Figura 4.19: Matriz Detalhe

4.3 Integração CAE e CAD – Desenvolvimento do aplicativo

A implantação do aplicativo para o Software Siemens NX, deu-se por meio da criação de uma biblioteca com vínculos dinâmicos (DLL), tendo todas as suas funções em um único arquivo compilado, que pode ser acessado pelo sistema CAD.

Para o desenvolvimento do aplicativo, foi utilizado à própria API existente no *software* Siemens NX. A decisão de uso deste sistema CAD deve-se à familiaridade e disponibilidade do mesmo para o seu uso. A escolha da utilização da linguagem VB.Net como linguagem de programação deve-se, a facilidade que o sistema CAD oferece quanto à disponibilidade de APIs e a utilização de um recurso que gera as macros referentes às operações realizadas no sistema CAD. A criação da biblioteca foi feita com o uso do software Microsoft Visual VB.Net, que é o compilador utilizado pela API.

A interface de programação do sistema Siemens NX, utilizando a ferramenta *User Interface Styler*, possui todas as funcionalidades do sistema CAD e permite criar a interface de comunicação com o usuário e a programação de funcionamento.

Este ambiente encontra-se dentro do próprio software e para utilizá-la basta ir à barra de ferramentas principal na opção "*Start*" e acessá-lo, conforme Figura 4.20.

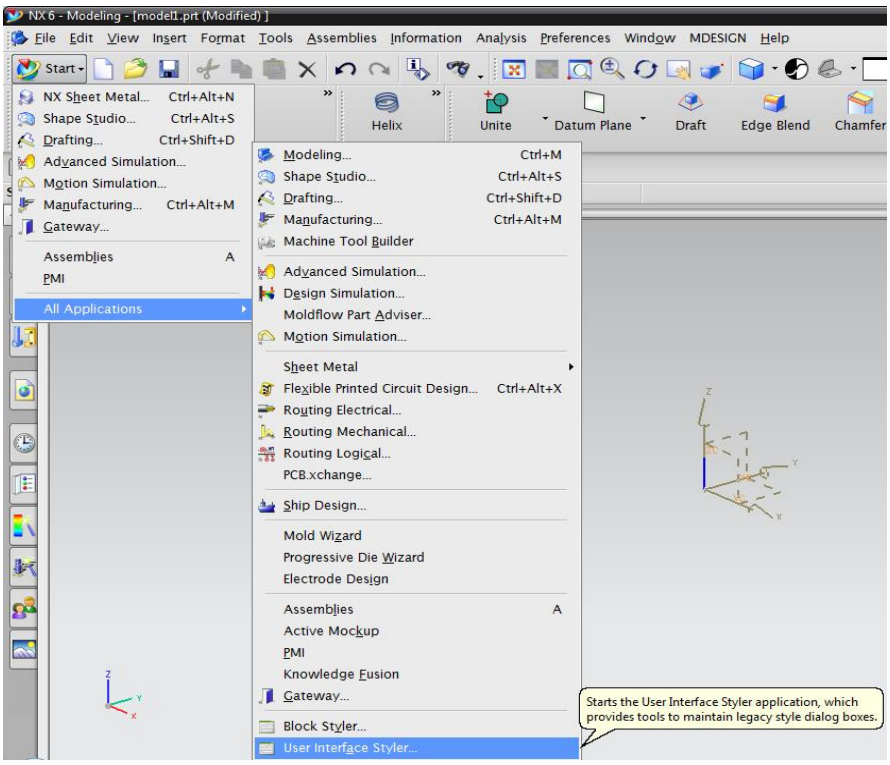


Figura 4.20: Acesso ao User Interface Styler.

Através desta opção, tem-se acesso à interface de programação do NX (apresentada na Figura 4.21). Esta interface contém diversas funções que podem ser agregadas e vinculadas posteriormente a alguma operação que o usuário desenvolva.

Conforme estas funções são selecionadas, o resultado visual é apresentado na interface (janela de diálogo) e fica visível na lista de funções. Com o editor, é possível alterar textos, mensagens e os nomes de chamada de cada função, que serão utilizados posteriormente na programação do aplicativo.

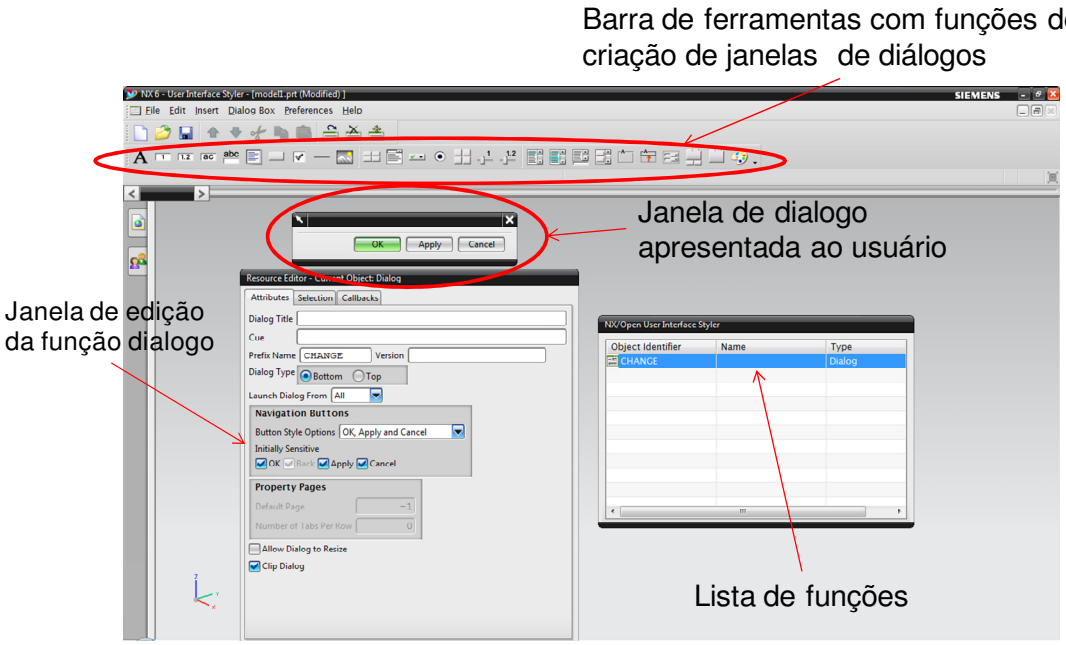


Figura 4.21: Janela de construção do User Interface Styler.

Após ter customizado uma interface que será utilizada pelo usuário e ao salvar os arquivos, deve-se escolher qual tipo de linguagem de programação será empregada para que os arquivos criados sejam compatíveis. Neste trabalho, foi utilizado a linguagem de programação VB.Net e com isso são gerados dois arquivos com extensões diferentes:

- *.dlg - é a janela de diálogo;
- *.vb - o código fonte da Janela de dialogo em linguagem VB.Net.

O sistema Siemens NX e sua interface de programação são responsáveis apenas pela geração da interface gráfica e seus respectivos códigos para a programação desejada. Uma vez feito isso, toda a parte de programação das funcionalidades de cada aplicativo deve ser feita utilizando o IDE da programação escolhida, neste caso Microsoft Visual Studio VB.Net.

4.3.1 Ambiente de programação VB.Net

Para se utilizar o código fonte gerado pelo sistema Siemens NX e implementar as funcionalidades que o aplicativo final terá, devem-se tomar alguns cuidados, pois o Siemens NX utiliza arquivos do tipo DLL para compilar o aplicativo e disponibilizá-lo em sua barra de funções.

No ambiente de programação VB.Net é necessário iniciar um novo projeto do tipo "*Class Library*", pois o mesmo serve para o desenvolvimento de arquivos do tipo DLL, como mostrado na Figura 4.22.

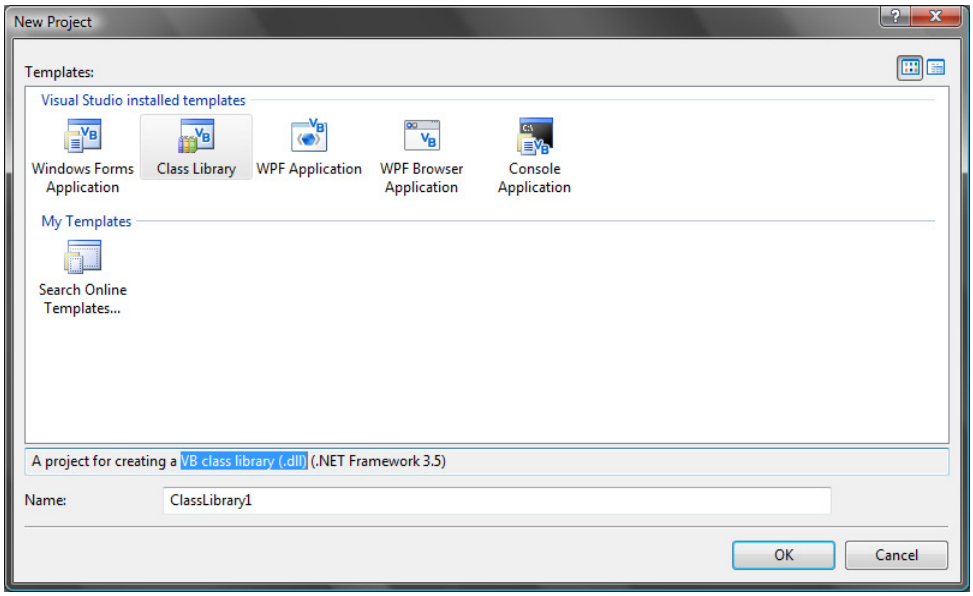


Figura 4.22: Tipo de projeto utilizado no Microsoft Visual Studio VB.Net

Após esta definição, na tela seguinte, é preciso copiar e colar o código fonte gerado pelo sistema Siemens NX para interface gráfica criada. No ambiente de trabalho do projeto, na opção "*Solution Explorer*", os arquivos gerados pelo API (extensões "*.vb") podem ser visualizados (Figura 4.23).

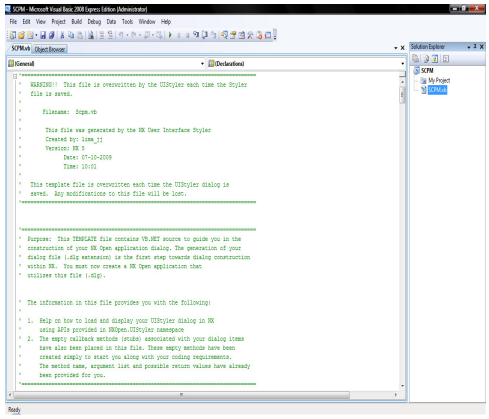


Figura 4.23: Arquivos no projeto.

Antes de compilar o programa são necessárias algumas configurações, para que haja a comunicação e identificação das bibliotecas do sistema CAD. Na opção "*Project->Properties*", deve-se acessar a aba *"References",* conforme Figura 4.24, onde se faz necessário adicionar os arquivos de referência

imprescindíveis para acessar funções e comandos do NX, das APIs do Windows e de manipulação de arquivos XML.

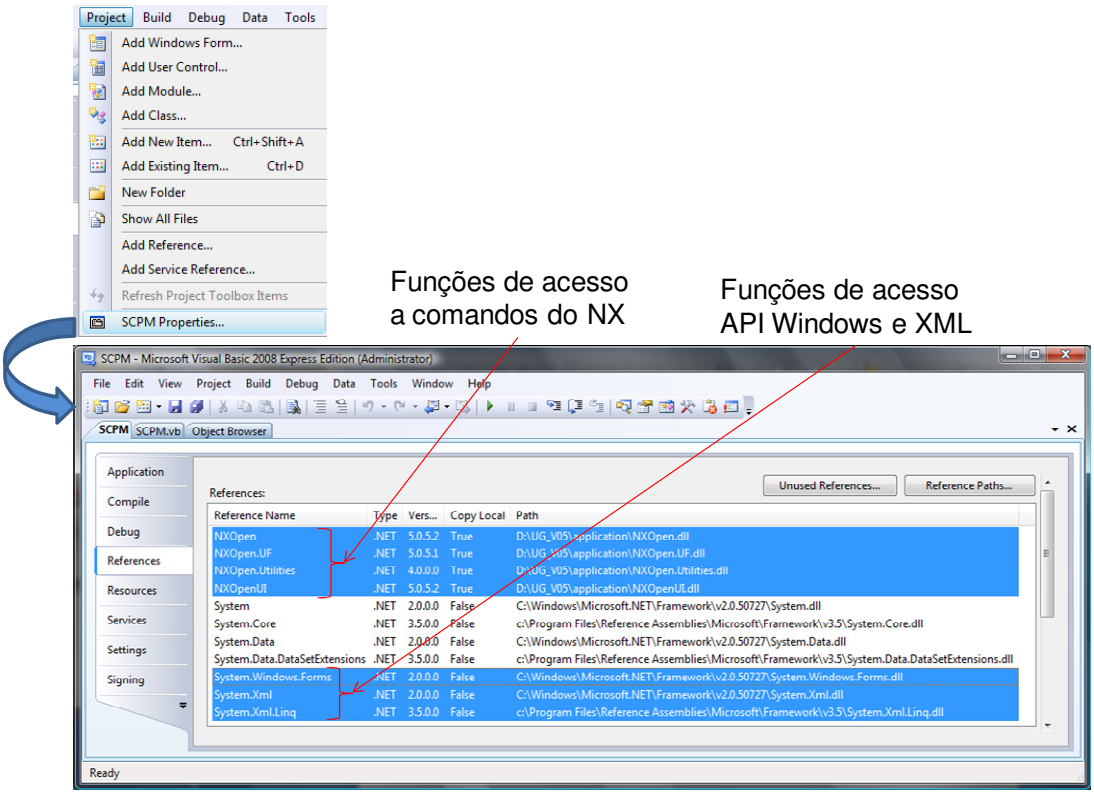


Figura 4.24: Configuração do ambiente de programação para acesso a funções e comandos específicos.

Na aba "*Compile*", determina-se o caminho para salvar os arquivos de saída. Esta pasta deve ser a pasta "*Application*", conforme Figura 4.25.

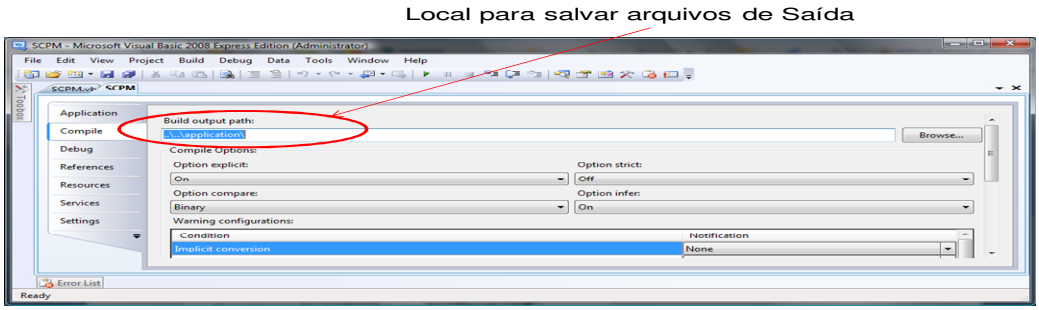


Figura 4.25: Configuração do caminho para pasta de saída dos dados

O próximo passo é dado pela disposição em que os arquivos se encontram para que a compilação ocorra corretamente. Duas novas pastas são criadas na pasta onde estão os arquivos gerados pelo NX: uma denominada "*Application*" e outra com o nome "*Startup*".

A criação da aba que será apresentada na barra de ferramentas do NX é gerada através de um texto existente no arquivo de extensão “*.vb”, como mostra a Figura 4.26. Este conteúdo é destinado especificamente para desenvolver o caminho de inclusão ao menu do CAD. O texto tem que ser copiado integralmente em um arquivo do tipo bloco de notas na pasta “*Startup*” e para o nome do arquivo deve-se utilizar o mesmo nome do projeto, com a extensão “*. men”.

Neste texto estão as informações da posição em que o aplicativo estará na barra de ferramentas e qual o formato que ele utilizará, como por exemplo, o efeito cascata. Todas estas informações podem ser alteradas de acordo com a necessidade do usuário.

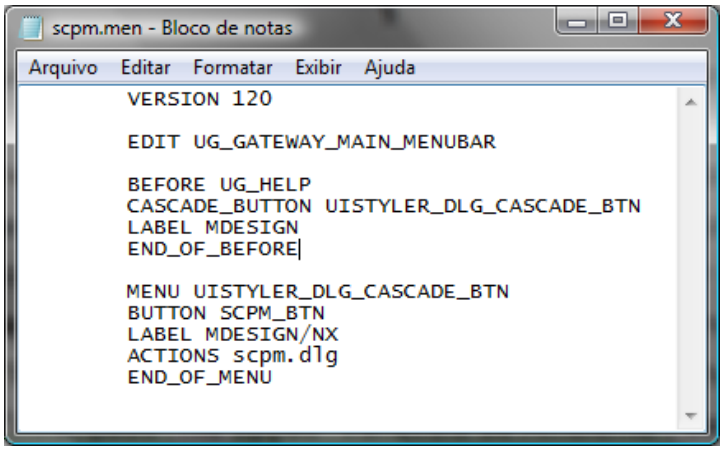


Figura 4.26: Texto contendo informações para a criação do menu.

Por fim, é necessário criar a base e os caminhos onde os dados serão requeridos para a compilação. O caminho que o sistema Siemens NX identificará serão as que contêm as informações do aplicativo – que precisa ser informado por meio de uma variável de ambiente – utilizando uma plataforma Windows, na opção de “Painel de Controle”, “Sistemas” e na opção “Configurações Avançadas do Sistema”. Ao Selecionar a opção “Variáveis de Ambiente”, deve-se criar uma nova variável com o texto “UGII_USER_DIR”, e o seu valor correspondente ao local em que se encontra o aplicativo (Figura 4.27).

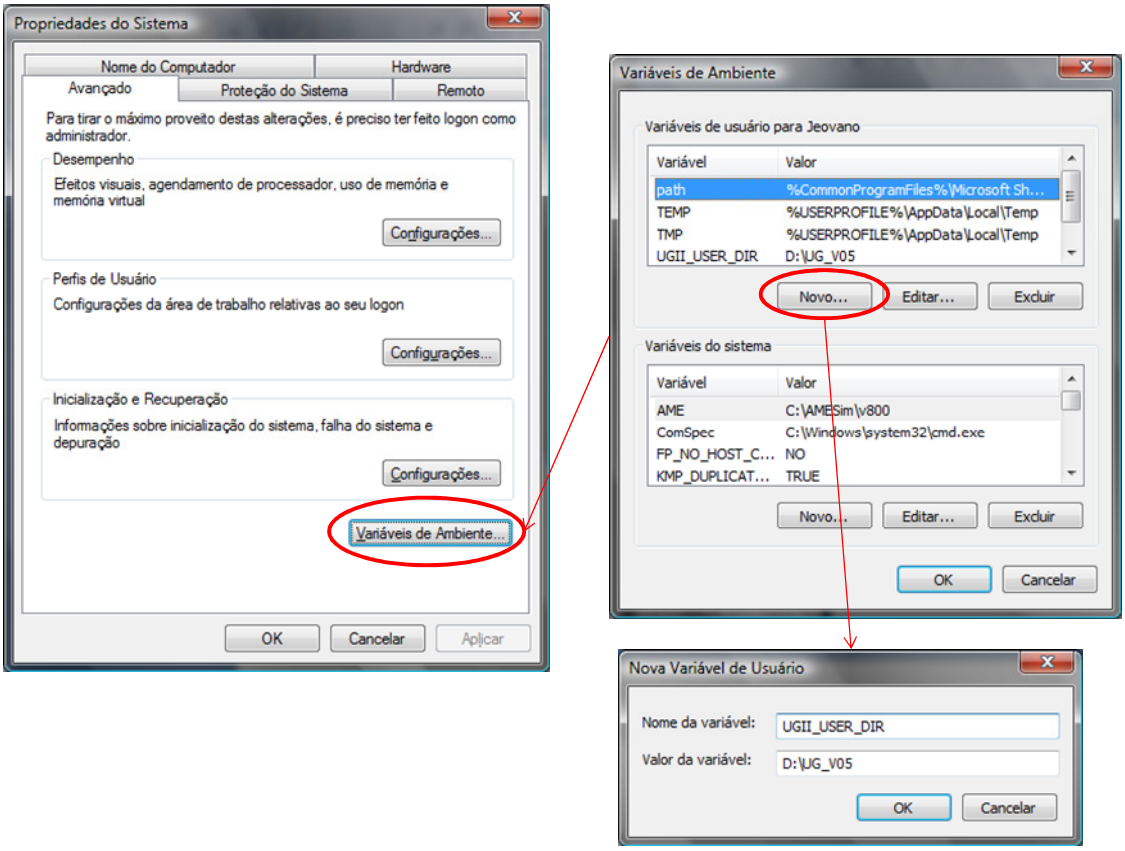


Figura 4.27: Configuração de variáveis de ambiente.

Ao se compilar o projeto, ele gerará o arquivo no formato DLL que será reconhecido pelo software CAD como uma nova biblioteca.

4.3.2 Geração da Geometria

Uma vez criada à matriz geométrica (ver tópico 4.2.2.2) se tem todos os dados para geração do modelo 3D, porém as funções de geração dos modelos foram implementadas no aplicativo com o auxilio da ferramenta *Journal*.

A ferramenta *Journal (*Figura 4.28*),* grava as operações executadas no software, salvando-as como código fonte vb.net, e servem como base para identificar quais funções o sistema Siemens NX utiliza para gerar seus modelos geométricos (cilindro, chaveta, rebaixos, etc.).

Para isso foi feito o seguinte procedimento de utilização da ferramenta *Journal*:

- Iniciar a gravação da sessão do sistema Siemens NX (Figura 4.28*);*
- Construir uma geometria simples (cilindro);
- Parar a gravação da sessão do sistema Siemens NX;

- Salvar a macro gerada VB.Net;
- Abrir o arquivo gerado e verificar os comandos e funções acessados pelo sistema Siemens NX.

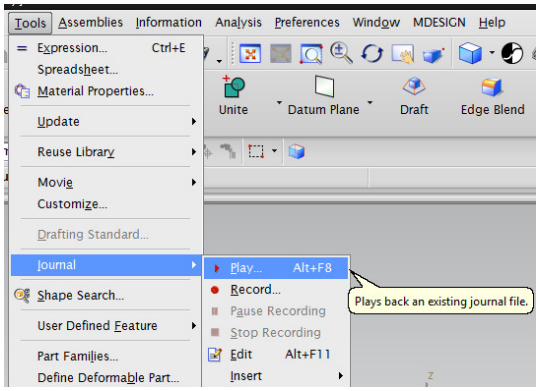


Figura 4.28: Recurso Journal do NX

Feito a gravação das operações de interesse (Cilindro, raios de arredondamento, Chavetas, rebaixo, entalhe), as mesmas foram implementadas. Para isso foram criadas 5 funções dentro de uma classe chamada *scpm* (Figura 4.29).

- Criarsecao() – Função de geração das seções e raios;
- CriarSketchrebaixo() – Função que gera croqui do rebaixo;
- Criarrebaixo() – Função que gera rebaixo utilizando o croqui gerado pela função CriarSketchrebaixo();
- CriarSketchChaveta() – Função que gera croqui da chaveta;
- CriarChaveta() – Função que gera chaveta utilizando o croqui gerado pela função CriarSketchChaveta().

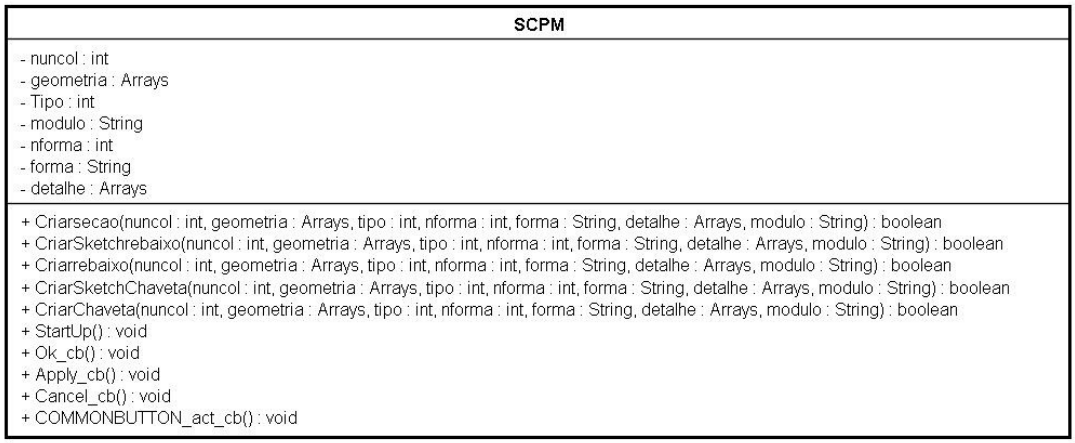


Figura 4.29: Classe scpm

A função Criarsecao() pode ser vista no diagrama de atividades da Figura 4.30 onde as seções são criadas utilizando os comandos `cylinderBuilder.Diameter` e `cylinderBuilder.Height` responsáveis pela criação dos objetos cilindros para que as seções não sejam criadas utilizando o mesmo ponto de referência é utilizado o comando `FindObject("EDGE * 3 * 1")`de define o lado do último cilindro criado como referencia para criação do próximo.

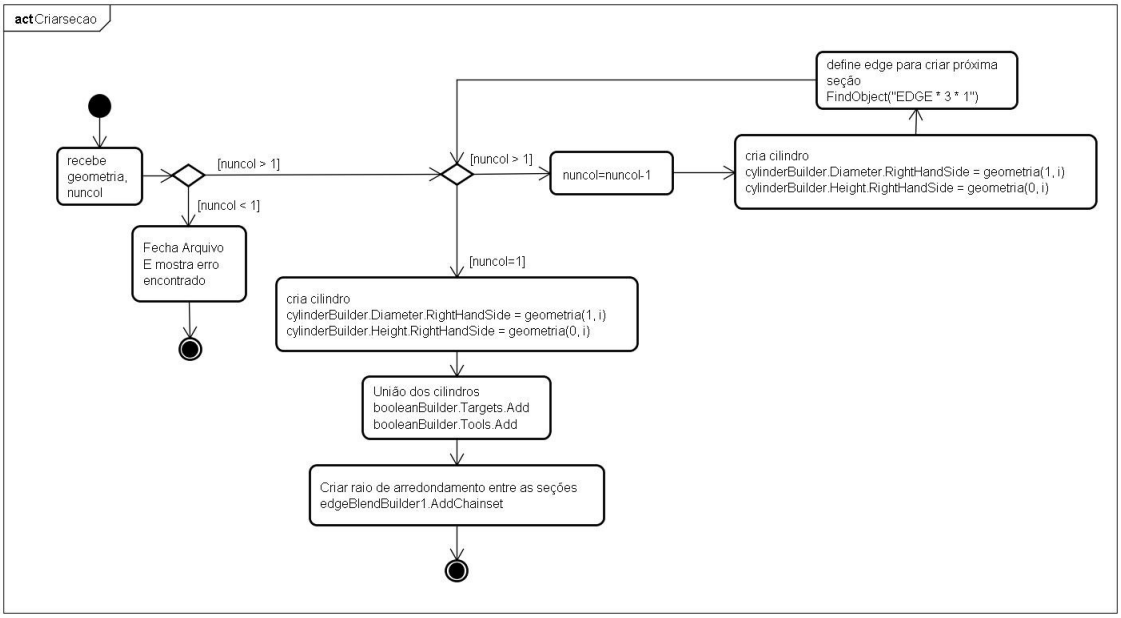


Figura 4.30: Diagrama de atividades da função Criarsecao().

Após a criar das seções é necessário unir as mesmas para isso utiliza-se o comando `booleanBuilder.Targets.Add` e `booleanBuilder.Tools.Add`. Com as seções unidas é possível criar os raios de arredondamento o comando `edgeBlendBuilder1.AddChainset` é utilizado para esta finalidade.

Para criação do rebaixo e da chaveta é necessário primeiro gerar um croqui, para isso foram desenvolvidos 2 funções, a CriarSketchrebaixo() e CriarSketchChaveta(). As duas funções possuem o mesmo diagrama de atividades conforme Figura 4.31.

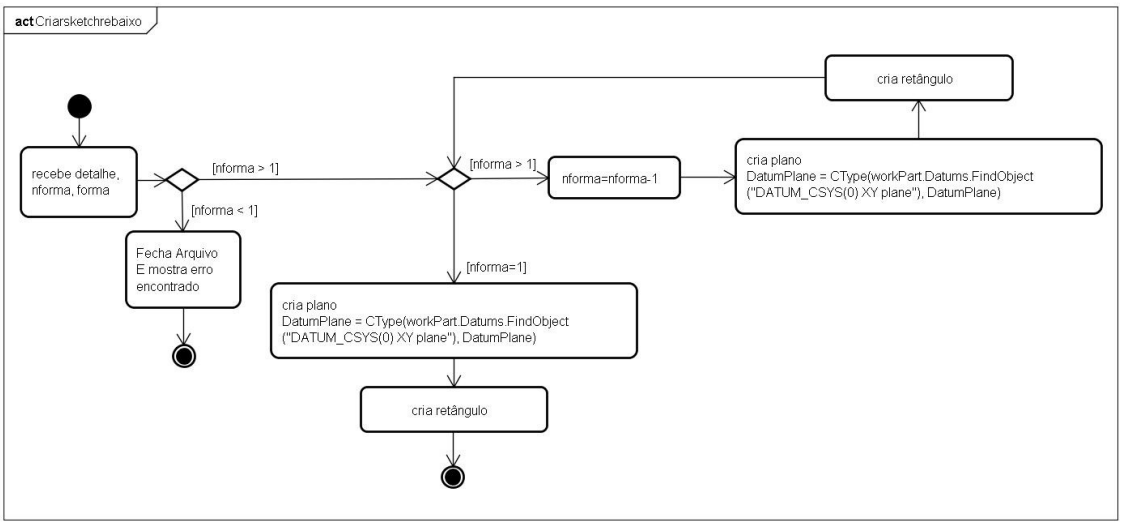


Figura 4.31: Diagrama de atividades das funções de criação de sketch

A função Criarrebaixo() utiliza a operação de revolução, já a função CriarChaveta() utiliza a operação e extrusão, as duas são dependentes da função de criação de croqui. O diagrama de atividades dessas funções pode ser visto na Figura 4.32.

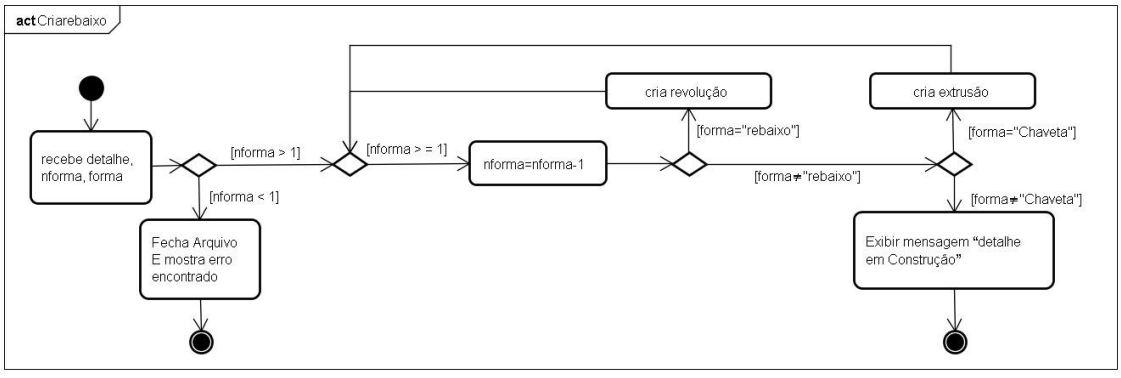


Figura 4.32: Diagrama de atividade das funções de Criarrebaixo() e CriarChaveta().

4.4 Interface com o usuário

Foi adotado como base o módulo de eixos, que contém várias opções que vão desde simples seções e raios de arredondamentos (Figura 4.33 a) a eixos com

rasgos de chavetas, entalhes, rebaixo para anéis elásticos e anéis trava (Figura 4.33 b).

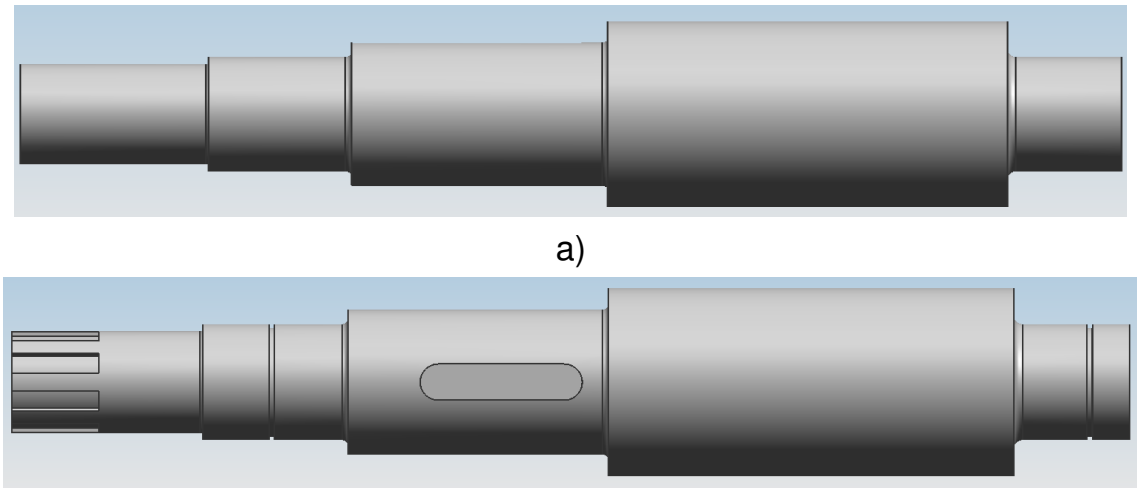


Figura 4.33: a) eixo simples b) eixo completo

Definido o módulo a ser utilizado para exemplificação da interface de integração CAE/CAD, uma interface simples e com acesso lógico às suas operações foi estruturada (Figura 4.34).

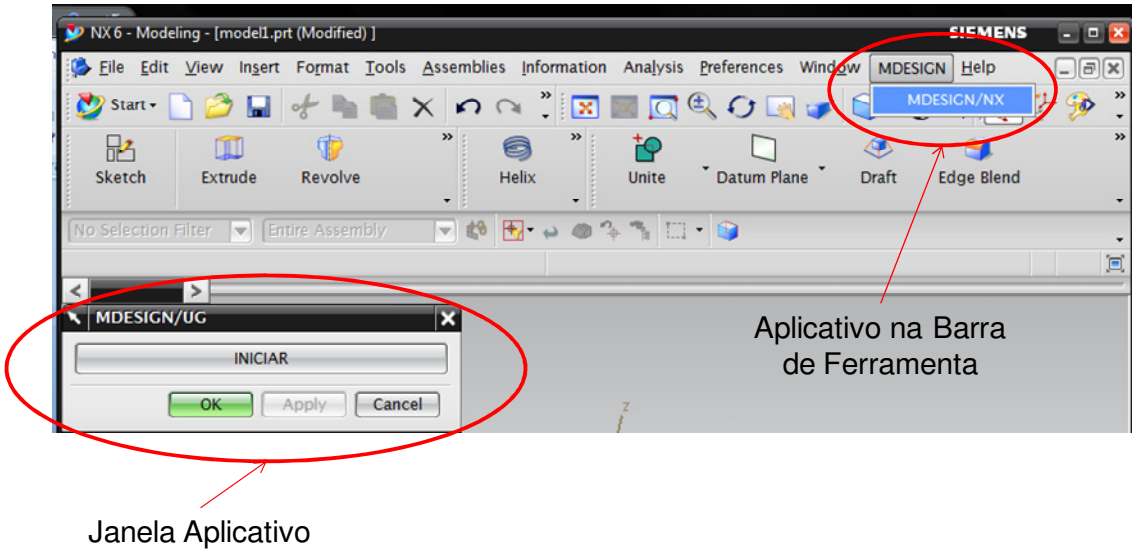


Figura 4.34: Interface com o usuário.

Esta estrutura baseia-se na própria barra de ferramentas do sistema CAD, onde o usuário seleciona uma aplicação e todas as funções relacionadas a ela são apresentadas. O esquema da Figura 4.35 ilustra a seqüência de passos necessários para utilização do aplicativo, incluindo as funções que dependem da interação do usuário, como as que são realizadas pelo sistema.

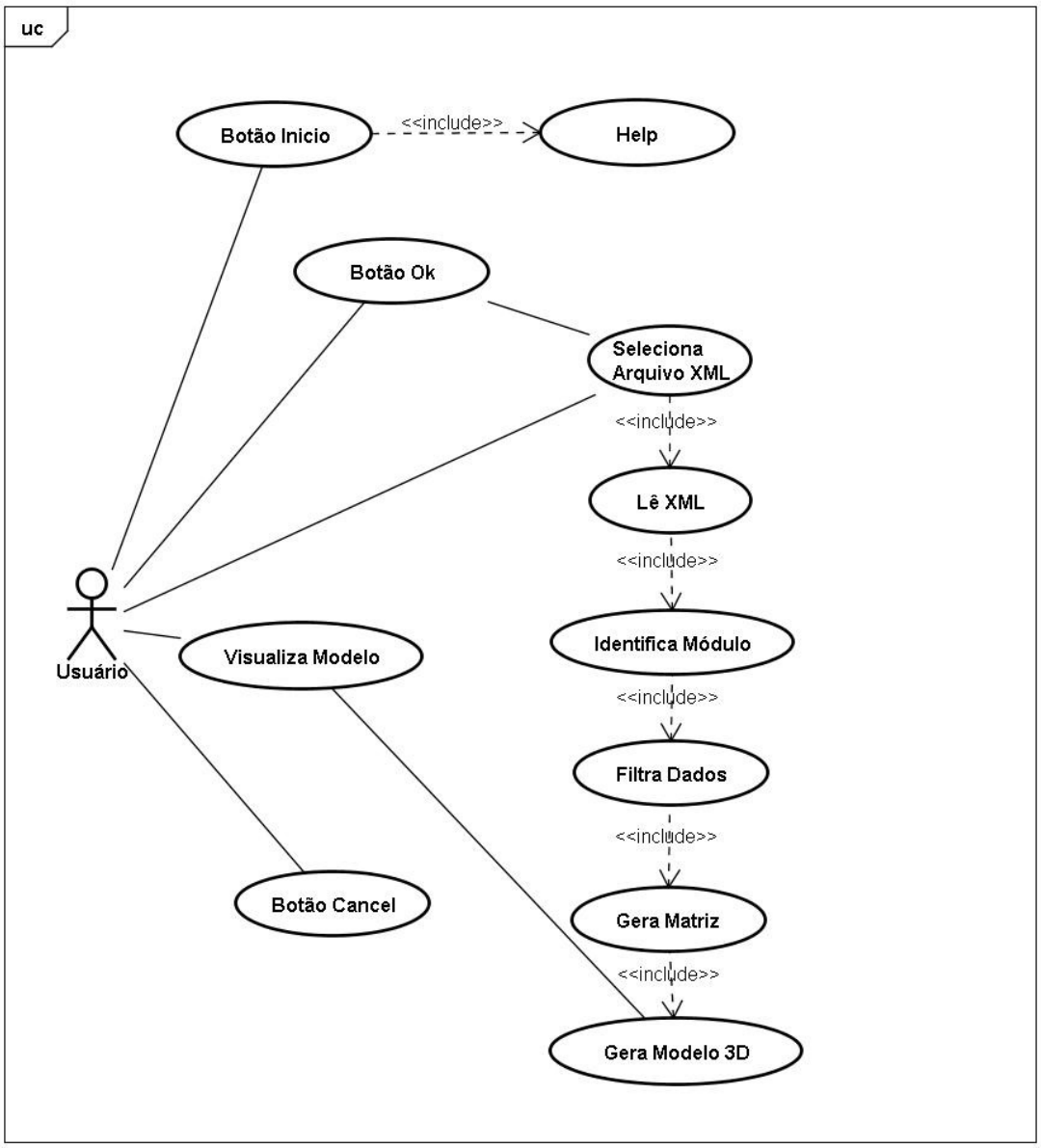


Figura 4.35: Seqüência de uso do aplicativo.

Para utilizar o aplicativo, o usuário deve selecionar na barra de opções principal do software, a opção “MDESIGN”, localizada à esquerda da opção “help”.

Após a seleção da opção criada para o aplicativo, uma nova opção estará disponível “MDESIGN/NX”. Com a seleção desta, uma nova janela é executada e com isso o usuário já se encontra dentro do aplicativo (Figura 4.36).

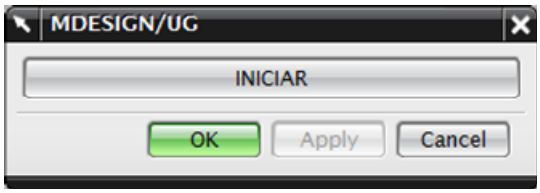


Figura 4.36: Janela de opções do aplicativo.

A opção “Iniciar” contém informações para execução do aplicativo, pois ela exibe a seguinte mensagem: "Para Iniciar o Programa é necessário utilizar o comando OK"

Quando selecionado a opção “OK”, o *browser* de seleção do arquivo do Windows é iniciado (conforme Figura 4.37) contendo nele o filtro para identificar apenas os arquivos XML.

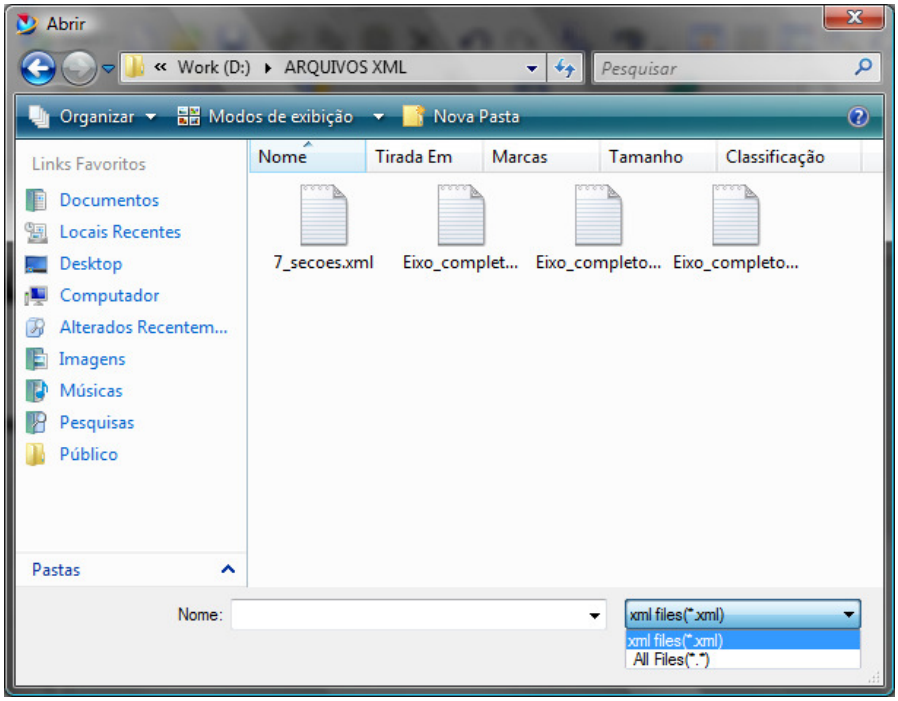


Figura 4.37: Janela de seleção do arquivo XML

Assim que o arquivo XML for selecionado, o aplicativo executará todas as linhas de comando e exibirá algumas mensagens (Figura 4.38) para verificação de qual operação estará sendo executada, como caminho do arquivo selecionado, nome do módulo ao qual o arquivo XML selecionado pertence (eixo, engrenagens, molas, etc.), no caso de eixos, quantas seções ele possui.

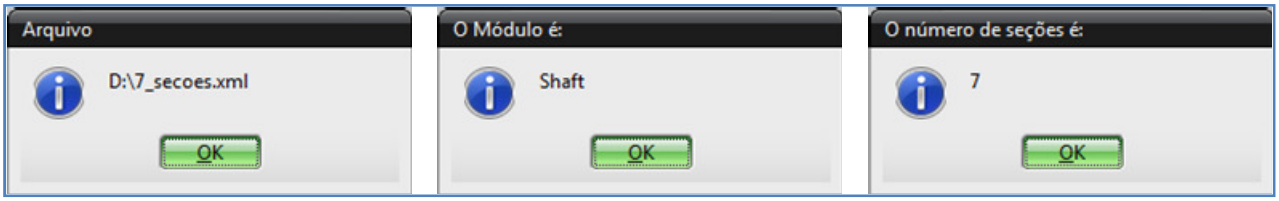


Figura 4.38: Mensagens de verificação.

Finalizando com a geração do modelo 3D conforme Figura 4.39, esse protótipo do aplicativo gera eixos de até 50 seções e insere os raios de arredondamento entre as seções, se necessário.

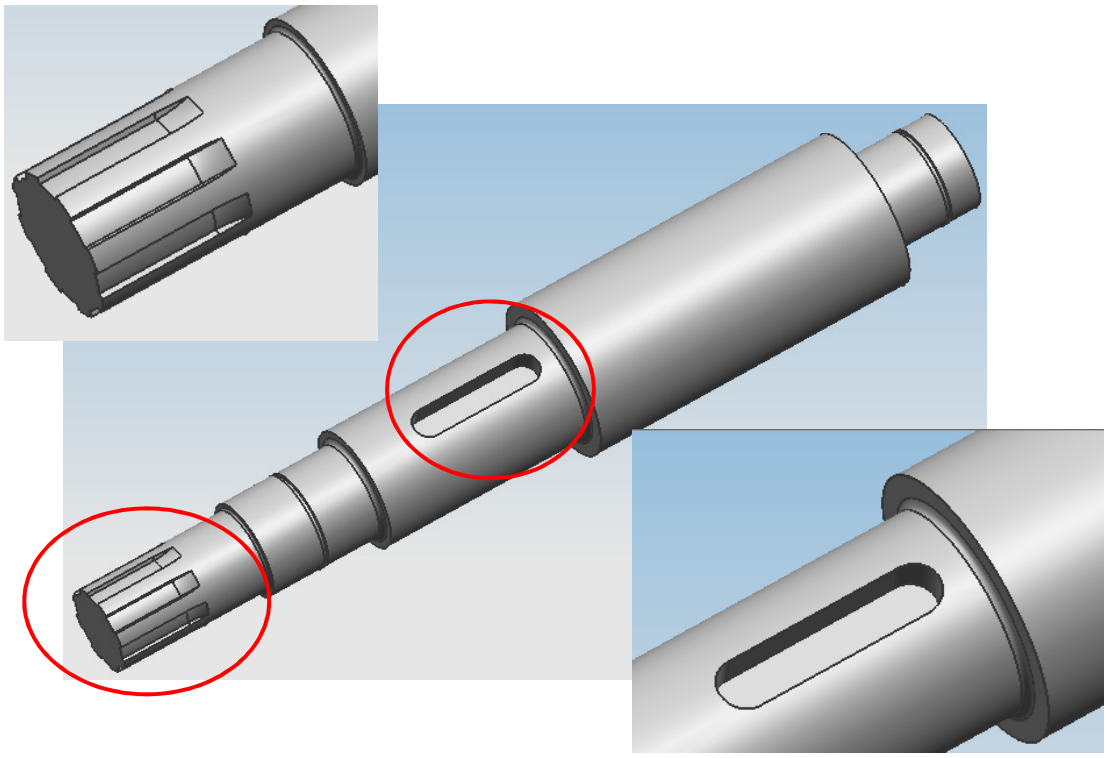


Figura 4.39: Exemplo de eixo gerado pelo aplicativo

Uma das vantagens deste aplicativo é que ele não gera apenas a geometria, mas toda a árvore de construção do modelo, podendo o mesmo ser editado facilmente bem como estabelecer uma padronização do método de modelamento (Figura 4.40).

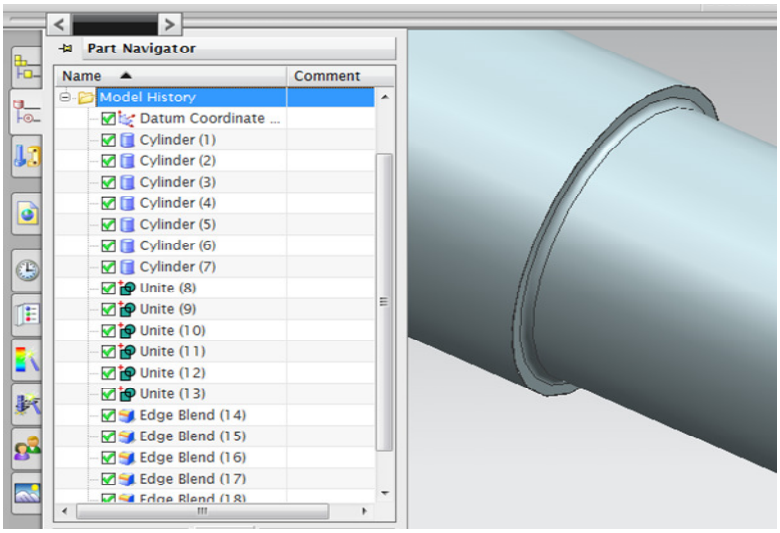


Figura 4.40: Árvore de construção gerada pelo aplicativo

4.5 Método de Avaliação

A avaliação teve por objetivo verificar os ganhos obtidos com a utilização do aplicativo, em relação ao método tradicional onde se tem um dimensionamento no MDESIGN e modelamento manual do elemento de máquina.

Para esta avaliação foram dimensionados no sistema CAE MDESIGN três modelos de eixos, para os quais foram gerados memoriais de cálculos, de onde são extraídas as informações geométricas para posterior modelamento 3D no sistema CAD Siemens NX.

Os modelos foram submetidos a três alunos: sendo eles um de mestrado com seis anos de experiência em sistemas CAD, e dois de graduação com um e três anos de experiência respectivamente, portanto possuindo domínio do sistema CAD Siemens NX.

Cada aluno utilizando as memórias de cálculo geradas no sistema CAE MDESIGN, construiu os modelos 3D de cada um dos três eixos. Estes mesmos modelos foram gerados utilizando o aplicativo desenvolvido neste trabalho.

Visando a comparação dos dois métodos de geração do modelo geométrico, foram avaliadas as seguintes características:

- Tempo de modelamento – característica fundamental para o desenvolvimento do produto;
- Número de elementos criados para geração do modelo – importante para identificação do modo como cada projetista modela um elemento de máquina;
- Facilidade de modificação do modelo gerado – defini o quão fácil será a modificação de um modelo quando a mesma for necessária;
- Erros no modelamento – identificar a influência do fator humano em erros durante o modelamento.

O tempo de modelamento considerou o tempo que os alunos levaram para modelar os eixos, desde o momento que eles receberam os memoriais de cálculo até terminarem o modelo 3D. Em alguns casos os alunos efetuaram cálculos e consultaram tabelas para encontrar parâmetros geométricos que não estão explícitos no memorial de cálculo.

O indicador “número de elementos” verifica qual método o aluno utilizou para gerar o modelo (*features* ou *sketch*). Com isso é possível verificar se o modelo construído é fácil ou difícil de ser modificado.

Todos os modelos gerados foram avaliados para verificar a existência de erros geométricos (medidas ou forma), comparando os mesmos com os requisitos do memorial de cálculo.

4.6 Resultados

Com base nos objetivos, os resultados mostram a comparação dos dois métodos utilizados para construção dos modelos geométricos dos 3 eixos, utilizando as características anteriormente apresentadas como avaliação.

4.6.1 Resultado do Modelo 1

O Modelo 1 consistiu de um eixo básico com 5 seções de diâmetros variáveis com raios de arredondamento entre as seções, conforme a Figura 4.41, a variação do tempo entre os alunos e resultante dos diferentes níveis de experiência de cada um possui. Por isso foi feito uma média dos tempos dos três alunos e o desvio da média, o mesmo foi feito para o aplicativo que também apresentou diferenças, devido ao aplicativo ter sido executado em diferentes *hardwares*. Com isso comparou-se os resultado do aplicativo com os dos alunos.

Modelo 1

	Alunos	Aplicativo
Tempo Médio (hh:mm:ss)	00:09:36	00:00:28
Amplitude (hh:mm:ss)	00:04:38	00:00:08

Figura 4.41: Resultados de tempo do Modelo 1

Nota-se na tabela que o tempo ganho com a utilização do aplicativo corresponde a cerca de 2000%.

Os outros resultados mostram a variação que existe quando uma mesma peça é feita por diferentes projetistas, pois cada um tem um nível de conhecimento do sistema e desenvolveu um método para construir seus modelos. Como é o

caso do número de elementos e a facilidade de modificação mostrados na Figura 4.42.

Alunos	Numero de Elementos	Fácilidade de modificação	Erro
1 - Manual	8	S	S
2 - Manual	14	S	N
3 - Manual	2	N	N
4 - Aplicativo	13	S	N

Figura 4.42: Resultados Modelo 1

O número de elemento deixa clara a diferença no modo como cada aluno construiu o eixo, deixando claro que existe um problema de padronização no método de modelamento. Isso impacta diretamente na facilidade de modificação do modelo geométrico, os dois primeiros alunos usaram *features* para construir seus modelos isso proporciona uma fácil modificação do modelo conforme já apresentado na revisão bibliográfica. Já o terceiro aluno utilizou um método em que todas às seções do eixo estão contidas em uma única operação (*Sketch*), causando dificuldades se houver a necessidade de modificá-lo.

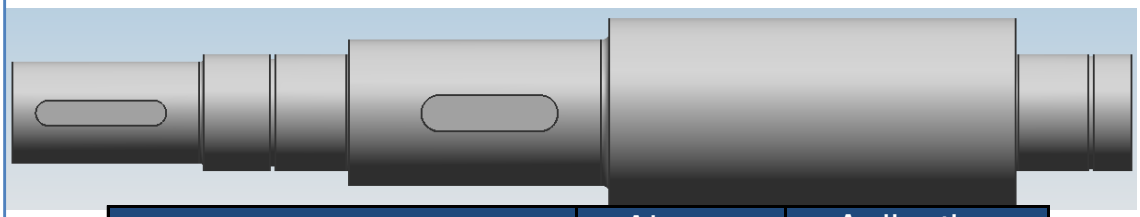
Pode-se observar que mesmo em uma peça simples é possível ocorrer erros quando se tem um processo manual, neste caso o fato dos alunos terem que identificarem e utilizarem informações de um memorial de cálculos.

4.6.2 Resultado dos Modelos 2 e 3

O Modelo 2 consistiu de um eixo com 5 seções de diâmetros variáveis, com raios de arredondamento entre as seções, com 2 chavetas e 2 rebaixos para anéis elásticos. O Modelo 3 possui 5 seções de diâmetros variáveis, com raios de arredondamento entre as seções, 1 chaveta, um entalhe e 2 rebaixos para anéis elásticos, conforme mostrado na Figura 4.43.

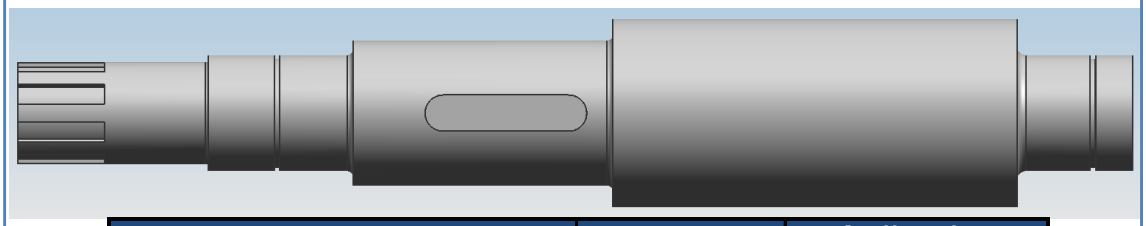
Assim como no modelo 1, houve variação do tempo entre os alunos e resultante dos diferentes níveis de experiência de cada um possui. E os métodos de comparação utilizados foram os mesmos. Com isso comparou-se os resultado do aplicativo com os dos alunos.

Modelo 2



	Alunos	Aplicativo
Tempo Médio (hh:mm:ss)	00:25:03	00:00:27
Amplitude (hh:mm:ss)	00:07:24	00:00:05

Modelo 3



	Alunos	Aplicativo
Tempo Médio (hh:mm:ss)	00:22:01	00:00:28
Amplitude (hh:mm:ss)	00:02:55	00:00:04

Figura 4.43: Resultados de tempo dos modelos 2 e 3

Nota-se na tabela que o tempo ganho com a utilização do aplicativo para os modelos 2 e 3 correspondem a cerca de 5000%.

Modelo 2

Alunos	Numero de Elementos	Fácilidade de modificação	Erro
1 - Manual	20	S	S
2 - Manual	5	N	S
3 - Manual	8	N	S
4 - Aplicativo		S	N

Modelo 3

Alunos	Numero de Elementos	Fácilidade de modificação	Erro
1 - Manual	12	S	S
2 - Manual	10	N	S
3 - Manual	8	N	S
4 - Aplicativo		S	N

Figura 4.44: Resultados dos modelos 2 e 3

O número de elemento nos dois modelos (Figura 4.44) novamente demonstra a diferença no modo como cada aluno construiu o eixo, relevando ainda mais o problema de padronização no método de modelamento.

Pode-se observar que quando se têm muitos detalhes no modelo fatalmente ocorrem erros, nestes dois caso alem de identificar as informações no memorial de cálculo foram necessários cálculos extras para encontrar informações geométricas que não estavam explicitas na mesma.

5 Conclusão

Nesta dissertação, a principal meta foi desenvolver um aplicativo para integração entre os sistemas CAE MDESIGN e CAD Siemens NX, para isso foi desenvolvido o aplicativo para o módulo de eixo e o mesmo foi avaliado. Com esse desenvolvimento pode-se destacar os seguintes aspectos:

- A utilização da linguagem XML como elo das informações entre os sistemas CAE/CAD, teve um bom desempenho, pois as informações do dimensionamento armazenados na mesma são de fácil manipulação (leitura, filtros, extração de dados);
- O uso da ferramenta *Journal* do NXOpen foi muito importante para compreender e gerar os códigos fontes de geração do modelo geométrico 3D, permitindo gravar os comandos e ações utilizados pelo usuário e facilitando a identificação das funções que o sistema CAD NX executa para isso;
- O aplicativo foi desenvolvido somente para o módulo de eixos, sendo possível a aplicação do mesmo conceito para os outros módulos de cálculo existentes no sistema MDESIGN;
- O Aplicativo é utilizado de maneira fácil e prática, pois o usuário, após gerar o dimensionamento do sistema CAE, só precisa abrir o arquivo salvo no formato XML, utilizando a interface desenvolvida no sistema CAD, e a geometria é gerada automaticamente;
- Além de gerar a geometria,o aplicativo também cria uma árvore de construção, que facilita a modificação do modelo gerado;
- Há um ganho quanto à acuracidade das informações, já que as mesmas são extraídas diretamente do arquivo gerado pelo sistema CAE, não tendo a necessidade de adicionar nada manualmente, como era feito no método tradicional;
- O ganho de tempo pôde ser observado na avaliação;
- Com a avaliação também se conclui que existe uma falta de padronização no modelamento geométrico, e com o aplicativo isso é facilmente possível.

De maneira geral, os objetivos do trabalho foram concluídos. Os conceitos aplicados para o desenvolvimento do aplicativo são genéricos. Assim sendo, pode ser aplicado a qualquer software que gere arquivos XML, como resposta ao dimensionamento e a outros módulos existentes no sistema CAE MDESIGN.

6 Bibliografias

[1] NAKAMURA, E. T.; Et al.: Utilização de ferramentas CAD/CAE/CAM no desenvolvimento de produtos eletroeletrônicos: vantagens e desafios. T & C Amazônia. Ano 1, n2, p.39-43, Junho 2003.

[2] MENDONÇA, D.R.; CAMARGO, R; SCANDIFFIO, I.: Integration of teh CAD/CAM/CAE System as a Tool to Lean Project Development. In: GCETE ´2005 - Global Congress on Engineering and Technology Education, 2005, Bertioga.

[3] CUNHA, G.D.: A Evolução dos Modos de Gestão do Desenvolvimento de Produtos, Produto & Produção, vol. 9, n. 2, p. 71-90, 2008.

[4] PAHL, G.; BEITZ, W. Engineering Design – A Systematic Approach. Londres: Springer-Verlag. 1996.

[5] BORGES, F. M; RODRIGUES, C. L. P.: Pontos passíveis de melhoria no método de projeto de produto de Pahl e Beitz, Gestão e Produção, V.17 Nº 2 - São Carlos – 2010.

[6] ALMEIDA, F. J.: Estudo e Escolha de Metodologia para o Projeto Conceitual, Ciência & Tecnologia, V. 8, Nº 16 – p. 31-42, 2000.

[7] DURKIN, J.: Expert Systems – design and development. New York, Prentice Hall, 1998.

[8] ULMANN, D.G. The Mechanical Design Process. New York: McGraw-Hill. 1997.

[9] SOBEK, D. K.: Transitions: From Conceptual Ideas to Detail Design. Engineering Education Annual Conference & Exposition. Montana. USA, 2005.

[10] SHIAU, J. Y; XIANGYANG, L.: Closed-Loop Design Change Control Workflow in Product Data Management for Conceptual Design Phase. International Journal of Intelligent Control and Systems, V 12, N 1, March 2007, 24-36

[11] FILHO, A. A.: Elementos Finitos - A Base da Tecnologia CAE Ed. Érica, S. Paulo, Brasil, 292 p, 2007.

[12] MEURER, A; ET AL.: CAE (Engenharia Auxiliada por Computador). Joinville, UDESC, 2003.

[13] REHG, J. A. Computer-integrated manufacturing. New Jersey, Prentice Hall, 1994.

[14] VAJPAYEE, S. K.: Principles of Computer-Integrated Manufacturing. Columbus, Prentice –Hall, 1995.

[15] BELYTSCHKO, T., JACOB, F.: Um primeiro curso de elementos finitos. Editora LTC 2009

[16] GARCIA, A. M.: Ajuste de Modelos Estruturais Aplicados em Problema de Contato. 113 f. Dissertação (Mestrado) – Faculdade de Engenharia de Ilha Solteira, Universidade Estadual Paulista “Julho de Mesquita Filho”, Ilha Solteira, 2006.

[17] ALMEIDA, F. J.: Implementação de Disciplina de Métodos Computacionais no Curso de Engenharia Mecânica, World Congress on Engineering and Technology Education, p. 1566-1570, São Paulo, Brasil, 2004

[18] SOBRINHO, A. S. C.: Introdução ao Método dos Elementos Finitos. Ed. Ciência Moderna, 2006.

[19] CHRISTOFORO, A. L.: Influência das irregularidades da forma em peças de madeira na determinação do modulo de elasticidade longitudinal, 156 f. Tese (Doutorado) – Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2007

[20] SOUZA, A. F. ; ULBRICH, C. B. L. . Engenharia Integrada por Computador e Sistemas CAD/CAM/CNC. Princípios e Aplicações. 1. ed. São Paulo: Artliber, 2009. v. 1. 332 p.

[21] COSTA, L. S. S.; CAULLINAUX, H.: Manufatura integrada por computador: Sistemas Integrados de Produção. Rio de Janeiro: Campus 1995

[22] AZEVEDO, Á. F. M.: Método dos Elementos Finitos. Cidade do Porto: Faculdade de Engenharia da Universidade do Porto, 2003.

[23] NIEMANN, J. Elementos de Máquinas, vol.1. 7ª ed; p.220, Edgard Blucher. 2002.

[24] NORTON, R. L.: Machine design: An Integrated Approach. New Jersey, Prentice – Hall. 3ª Ed. 984p. 2005.

[25] SHIGLEY, J.G. Mechanical Engineering Design. Oakland: Mc-Graw-Hill. 1981

[26] SPOTTS, M.F; SHOUP,T.E.: Design of Machine Elements . 7.ed. 1998, Prentice Hall.

[27] SILVEIRA, Z. C.: Desenvolvimento de um Sistema Computacional de Auxilio ao Cálculo e Desenho de Elementos de Máquinas. São Carlos, SP. 180p. Dissertação (Mestrado). Escola de Engenharia de São Carlos. Universidade de São Paulo. 1998.

[28] SCHÜTZER, K.: Desenvolvimento de um sistema de projeto de engrenagens auxiliado por computado. São Carlos, SP. 236p. Dissertação (Mestrado). Escola de Engenharia de São Carlos. Universidade de São Paulo. 1988.

[29] BESANT, C.B.: CAD/CAM Projeto e fabricação auxiliada por computador . Rio de Janeiro, Campus, 1983.

[30] REMBOLD, U; NNAJI, B. O.; STORR, A. CAD: It's Role in Manufacturing. In: Computer Integrated Manufacturing and Engineering. Wokingham: Addison Wesley, 1993.

[31] SPECK, H. J. Proposta de método para facilitar a mudança de técnicas de projetos: da prancheta à modelagem sólida (CAD) para empresas de engenharia de pequeno e médio porte. 2005. 185 f. Tese (Doutorado em Engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis.

[32] SOLINHO, J. L. G.: A Indústria Mecânica e a Revolução do processo de Projeto. CADware Technology **,** São Paulo, ano 2, n. 8, p. 31-33, 1998.

[33] FERREIRA, A. B.: Conceitos e Aplicações em Projetos Mecânicos e Critério para a Seleção e Utilização em Engenharia. 102 f. Dissertação (Mestrado) – Escola Politécnica, Universidade de São Paulo, São Paulo, 1990.

[34] VALENTIN, H. R.; CORREIA, R. Q.: Sistema CAD: evolução e tendências. 40 f. Monografia (Especialização) – Pós Graduação "Lato-sensu" Especialização em Análise de Sistema, Centro Universitário de Belo Horizonte, Belo Horizonte, 2002.

[35] FRANCESCONI, T.: Proposta Metodológica para Modelagem Geométrica de Imagens Médicas. 105 f. Dissertação (Mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2008.

[36] FOGGIATTO, J.A; VOLPATO, N.; BONTORIN, A. C. B.: Recomendações para Modelamento em Sistemas CAD 3D, In: 4º Congresso Brasileiro de Engenharia da Fabricação, COBEF, 2007, São Pedro SP.

[37] MCMAHON, C.; BROWNE, J.; "CAD CAM: Principles, Practice and Manufacturing Management", Addison Wesley Longman, 2nd edition. 1998.

[38] ZEID, I.: Mastering CAD/CAM. New York: McGraw Hill, 2005, 962p.

[39] ALVES, R.: Técnicas de CAD. 2002. 58f. – Apostila Departamento de Expressão Gráfica, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

[40] SPECK, H. J. Avaliação Comparativa das Metodologias Utilizadas em Programas de Modelagem Sólida. 2001. 203 f. Dissertação (Mestrado em engenharia de Produção) - Universidade Federal de Santa Catarina, Florianópolis.

[41] SHAH, J.J.; ROGERS, M. T.: Expert Form Feature Modelling Shell. Computer Aided Design. V.20, n.9, p.515-524, 1998.

[42] LEE, S. H.: A CAD-CAE Integration Approach Using Feature-based Multi-resolution and Multi-abstraction Modeling Techniques. Computer-Aided Design**,** V 37, n 9, Agosto 2005, p. 941-955

[43] SCHÜTZER, K.: Integrierte Konstruktionsumgebung auf der Basis von Fertigungsfeatures. München: Hanser Verlag, 1995

[44] SHAHIM, T.M.M; Feature-Based Design – An Overview, Computer-Aided Design & Applications, V 5, n 5, 2008, p. 639-653.

[45] EIGNER, M; HANDSCHUH, S; GERHARDT, F.: Concept to Enrichen Lightweight, Neutral Data Formats with CAD-based Feature Technology. Computer-Aided Design & Applications, V 7, n 1 2010, p. 89-99.

[46] FIGUEIRA, R. J. C. M.: CAD/CAM/CAE/CIM. 2003. 123 f. Licenciatura em Computadores e Sistemas – Instituto Superior de Engenharia do porto, Porto, 2003.

[47] BASAK, H.; GÜLESIN, M.: A Feature Based Parametric Design Program and Expert System for Design, Mathematical and Computational Applications, Vol. 9, No. 3, pp. 359-370, 2004.

[48] KERRY, H. T.: Planejamento de processo automático para peças paramétricas. 1997. 173 f. Dissertação (Mestrado) - Escola de Engenharia de São Carlos, Universidade de São Paulo.

[49] PERES, M. P.; HAYAMA, A. O. F.; VELASCO, A. D.: A Parametrização e a Engenharia. Ghafica. 2007. Curitiba.

[50] FERREIRA, P. H. Projeto e Otimização de Árvore de Manivelava. 2008. 144 f. Dissertação (Mestrado em engenharia de Mecânica) – Escola Politécnica da Universidade de São Paulo, São Paulo.

[51] MARCHAL, B.: XML Conceitos e Aplicações. 1. ed. São Paulo: Editora Berkeley, 2000

[52] GONZÁLEZ, M; ALVAREZ, E; GARCIA DE JALÓN, J.: Mechml: un Nuevo Lenguaje Basado en XML para la Descripción de Sistemas Mecánicos Multi-Cuerpo. Espanha: SEMNI, 2002.

[53] World Wide Web Consortium (W3C). 2001. “Mathematical Markup Language (MathML)”,<http://www.w3.org/Math>.

[54] P. Murray-Rust and H. Rzepa. 2002. “Chemical Markup Language (CML)”, <http://www.xml-cml.org>.

[55] Open GIS Consortium Inc., 2000. “Geography Markup Language (GML)”, <http://opengis.net/gml/>.

[56] SENTHIL KUMAR; R, GOKUL NATH, M; RAAGHUL, U.: An Approach for UML based Scenario Oriented Slicing. International Journal of Recent Trends in Engineering, Vol. 1, n. 2, May 2009, p. 141-143

[57] PASSOS, R. A. N. O. Uma Aplicação Baseada em Ontologia para Integração Semântica em e-Business. 2006. Dissertação (Mestrado em Ciências da Computação) - Universidade Federal de Pernambuco.

[58] KÜHL, M. G.: Acesso Remoto de um Braço Eletrônico via Web Service. 79 f. Monografia (Especialização) – Centro Federal de Educação Tecnológica do Paraná, Pato Branco - PR, 2005.

[59] ZHONG, X.; XIAOYAN LI, J. Y.: Study on XML Based Visualization Interface for Engineering Data. vol. 1, pp.1096-1099, 2009 First International Workshop on Education Technology and Computer Science, 2009

[60] ANDERSON, R. et al. Professional XML. 1. ed. Rio de Janeiro: Ciência Moderna, 2001. 1266p

[61] KOSAVINTA, S.; KANONGCHAIYOS, P.; JINUNTUYA, P.: Integration of CAD Software with DSS for Engineering and Architectural Project Design, Computer-Aided Design & Applications, 4(1-4), 2007, 467-476.

[62] SHEHAB, E. M.; ABDALLA, H. S.: A cost-effective knowledge-based reasoning system for design for automation, Proc. IMechE, 220 (Part B: J. Engineering Manufacture), 2006, 729-743.

[63] KING, M. L.; FISHER, M. J.; JENSEN, C. G.: A CAD-Centric Approach to CFD Analysis with Discrete Features, Computer-Aided Design & Applications, Vol. 3, Nº. 1-4, 2006, pp 279-288.

[64] KENWORTHY, T.: CAD-CENTRIC Dynamic Workflow Creation, 2009. 91 p. Dissertação (Mestrado) - Brigham Young University, Provo - Utah.

[65]KAO, Y.C; CHENG, H.Y; SHE, C.H: Development of an integrated CAD/CAE/CAM system on taper-tipped thread-rolling die-plates. **Journal of Materials Processing Technology**, v.177, n.1-3, p.98-103, 2006.

[66] THEIS, K; TRAUTMANN. T.: Avoiding the Traps of CAD/CAE Integration. **Product Data Journal,** V 14, n 2, 2007, p. 29-31.

[67] BALAKRISHNA, A.; Et al.: Integration of CAD/CAM/CAE in Product Development System Using STEP/XML. **Concurrent Engineering**. V. 14, n2, p.121-128, Junho 2006.

[68] LUCENA, H. N.: Geração de estratégias de medição de superfícies complexas em sistema CAD para máquinas de medir por coordenadas. 135 f. Dissertação (Mestrado) – Universidade Metodista de Piracicaba, Santa Bárbara D'Oeste, 2009.

[69] HOGGE, D. Integrating Commercial CAx Software to Perform Multidisciplinary Design Optimization. 2002. Thesis, Brigham Young University, Provo - Utah.

[70] DING, S.; et al. The implementation of adaptive isoplanar tool path generation for the machining of free-form surfaces. International Journal Advanced Manufacturing Technology, v. 26, p. 852–860, 2005.

[71] LU, Z.; CHEN, D.; CHENG, Y. Effective Distributed Collaborative Design FrameBased on UG Software. 2nd International Conference on Pervasive Computing and Applications, p. 158-161, 2007.

[72] DE POLI, G. P.: Robust Multi-Disciplinary Design of Components Concluding Report. Value Improvement through a Virtual Aeronautical Collaborative Enterprise - VIVACE, 2007.

[73] ROZVANY, G.I.N.; ZHOU, M.; BIRKER, T. Generalized shape optimization without homogenization. Structural and Multidisciplinary Optimization. V. 4, p. 250–252, 1992.

[74] RODGERS, R.: NX Open New and Cool, PLM World 06, 2006, Long Beach – US.

Anexo 1

Exemplo de um arquivo XML gerado pelo sistema CAE MDESIGN.

```
<?xml version="1.0" encoding="windows-1252" standalone="yes" ?>
- <Document Signature="BSPMEDAT" Version="1.0" CalcID="352" CalcName="Shaft">
  <Header Date="1.11.2004" User="Standard" Customer="" Project=""
Comment="Mitgeliefertes Beispiel, TEDATA GmbH" />
- <Data>
  <Parameter Name="Hypothese" Type="combo" Index="2" Value="Design Changing Force
Hypothesis" />
  <Parameter Name="kappa_zd" Type="double" Value="0" />
  <Parameter Name="kappa_b" Type="double" Value="-1" />
  <Parameter Name="kappa_t" Type="double" Value="-1" />
  <Parameter Name="Rechenmethod" Type="combo" Index="1" Value="Roloff/Matek" />
  <Parameter Name="DIN_Bp" Type="combo" Index="1" Value="Case 1 (constant mean stress
sm)" />
  <Parameter Name="DIN_Of" Type="combo" Index="1" Value="none Kv = 1.0" />
  <Parameter Name="DIN_kz" Type="double" Value="1" />
  <Parameter Name="DIN_kb" Type="double" Value="1" />
  <Parameter Name="DIN_kt" Type="double" Value="1" />
  <Parameter Name="Werkstoff" Type="string" Value="S235JR" />
  <Parameter Name="Wrkstff_DINnum" Type="string" Value="1.0037" />
  <Parameter Name="Wrkstff_Rm" Type="double" Value="340" MeasureBase="N/mm²"
Measure="N/mm²" />
  <Parameter Name="Wrkstff_Re" Type="double" Value="235" MeasureBase="N/mm²"
Measure="N/mm²" />
  <Parameter Name="Wrkstff_E" Type="double" Value="215000" MeasureBase="N/mm²"
Measure="N/mm²" />
  <Parameter Name="Wrkstff_G" Type="double" Value="83000" MeasureBase="N/mm²"
Measure="N/mm²" />
  <Parameter Name="Wrkstff_rho" Type="double" Value="7.85" MeasureBase="kg/dm³"
Measure="kg/dm³" />
  <Parameter Name="Wrkstff_ny" Type="double" Value="0.3" />
  <Parameter Name="Wrkstff_x" Type="double" Value="500" MeasureBase="mm" Measure="mm" />
  <Parameter Name="Wrkstff_n" Type="double" Value="1000" MeasureBase="1/min"
Measure="1/min" />
  <Parameter Name="Wrkstff_Flag" Type="boolean" Value="0" />
- <Parameter Name="GEO" Type="list" Columns="10" Rows="7">
  <Column Name="GEO_l" Type="double" MeasureBase="mm" Measure="mm"
Value="150,200,100,150,100,200,150" />
  <Column Name="GEO_Da_l" Type="double" MeasureBase="mm" Measure="mm"
Value="50.5,60,70,80,70,60,50.5" />
  <Column Name="GEO_Di_l" Type="double" MeasureBase="mm" Measure="mm"
Value="0,0,0,0,0,0,0" />
  <Column Name="GEO_Da_r" Type="double" MeasureBase="mm" Measure="mm"
Value="50.5,60,70,80,70,60,50.5" />
  <Column Name="GEO_Di_r" Type="double" MeasureBase="mm" Measure="mm"
Value="0,0,0,0,0,0,0" />
  <Column Name="GEO_r" Type="double" MeasureBase="mm" Measure="mm" Value="3,3,2,2,3,3,0"
/>
  <Column Name="GEO_Rz" Type="double" MeasureBase="181;m" Measure="181;m"
Value="5,5,25,25,25,25,25" />
  <Column Name="Beta_kzd" Type="double" Value="10,0,0,0,0,0,0" />
  <Column Name="Beta_kb" Type="double" Value="0,0,0,0,0,0,0" />
  <Column Name="Beta_kt" Type="double" Value="0,0,0,0,0,0,0" />
  </Parameter>
- <Parameter Name="Lager" Type="list" Columns="7" Rows="2">
  <Column Name="Lager_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value="0,1030"
/>
  <Column Name="Lager_cx" Type="double" Value="-1,0" />
  <Column Name="Lager_cy" Type="double" Value="-1,-1" />
  <Column Name="Lager_cz" Type="double" Value="-1,-1" />
  <Column Name="Lager_Tx" Type="double" Value="0,0" />
  <Column Name="Lager_Ty" Type="double" Value="0,0" />
  <Column Name="Lager_Tz" Type="double" Value="0,0" />
  </Parameter>
- <Parameter Name="Massen" Type="list" Columns="3" Rows="0">
  <Column Name="Mass_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value="" />
  <Column Name="Mass_tragmom_J" Type="double" MeasureBase="kgm²" Measure="kgm²" Value=""
/>
  <Column Name="Mass_zusatz_m" Type="double" MeasureBase="kg" Measure="kg" Value="" />
  </Parameter>
- <Parameter Name="Axial_krft" Type="list" Columns="4" Rows="0">
```

```
  <Column Name="Axial_Fax_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value="" />
  <Column Name="Axial_Fax" Type="double" MeasureBase="N" Measure="N" Value="" />
  <Column Name="Axial_Fax_radius" Type="double" MeasureBase="mm" Measure="mm" Value=""
/>
  <Column Name="Axial_Fax_winkel" Type="double" MeasureBase="°" Measure="°" Value="" />
  </Parameter>
- <Parameter Name="Radial_krft" Type="list" Columns="4" Rows="1">
  <Column Name="Radial_Fr_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value="515"
/>
  <Column Name="Radial_Fr" Type="double" MeasureBase="N,N/mm" Measure="N,N/mm" Value="-
20" />
  <Column Name="Radial_Fr_lange" Type="double" MeasureBase="mm" Measure="mm" Value="0"
/>
  <Column Name="Radial_Fr_winkel" Type="double" MeasureBase="°" Measure="°" Value="0" />
  </Parameter>
- <Parameter Name="Biege_mom" Type="list" Columns="3" Rows="0">
  <Column Name="Biegemom_Mb_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value=""
/>
  <Column Name="Biegemom_Mb" Type="double" MeasureBase="Nm" Measure="Nm" Value="" />
  <Column Name="Biegemom_Mb_winkel" Type="double" MeasureBase="°" Measure="°" Value=""
/>
  </Parameter>
  <Parameter Name="Betriebstemperatur" Type="double" Value="20" MeasureBase="°C"
Measure="°C" />
- <Parameter Name="Torsion" Type="list" Columns="3" Rows="0">
  <Column Name="Torsmom_T_pos_x" Type="double" MeasureBase="mm" Measure="mm" Value="" />
  <Column Name="Torsmom_T" Type="double" MeasureBase="Nm,Nm/mm" Measure="Nm,Nm/mm"
Value="" />
  <Column Name="Torsmom_T_lange" Type="double" MeasureBase="mm" Measure="mm" Value="" />
  </Parameter>
  <Result Name="out_dlg" Type="double" Key="1" Value="1050.00" />
  <Result Name="@_2632" Type="void" Key="1" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dm" Type="double" Key="2" Value="25.56" />
  <Result Name="@_2633" Type="void" Key="2" MeasureBase="kg" Measure="kg" />
  <Result Name="out_dJ" Type="double" Key="3" Value="0.013935" />
  <Result Name="@_2634" Type="void" Key="3" MeasureBase="kgm²" Measure="kgm²" />
  <Result Name="out_dXc" Type="double" Key="4" Value="525.000" />
  <Result Name="@_2635" Type="void" Key="4" MeasureBase="mm" Measure="mm" />
  <Result Name="@_1750" Type="string" Key="0" Value="1" />
  <Result Name="@_1751" Type="string" Key="0" Value="2" />
  <Result Name="@_2671" Type="void" Key="140" MeasureBase="N" Measure="N" />
  <Result Name="out_dFry[1]" Type="double" Key="140" Value="10.0" />
  <Result Name="out_dFry[2]" Type="double" Key="140" Value="10.0" />
  <Result Name="@_2676" Type="void" Key="141" MeasureBase="N" Measure="N" />
  <Result Name="out_dFrz[1]" Type="double" Key="141" Value="0.0" />
  <Result Name="out_dFrz[2]" Type="double" Key="141" Value="0.0" />
  <Result Name="@_2681" Type="void" Key="142" MeasureBase="N" Measure="N" />
  <Result Name="out_dFr[1]" Type="double" Key="142" Value="10.0" />
  <Result Name="out_dFr[2]" Type="double" Key="142" Value="10.0" />
  <Result Name="@_2686" Type="void" Key="143" MeasureBase="N" Measure="N" />
  <Result Name="out_dFax[1]" Type="double" Key="143" Value="0.0" />
  <Result Name="out_dFax[2]" Type="double" Key="143" Value="0.0" />
  <Result Name="@_2691" Type="void" Key="44" MeasureBase="°" Measure="°" />
  <Result Name="out_dteta[1]" Type="double" Key="44" Value="0.000426" />
  <Result Name="out_dteta[2]" Type="double" Key="44" Value="0.000412" />
  <Result Name="out_dMbmax" Type="double" Key="6" Value="5.2" />
  <Result Name="@_2637" Type="void" Key="6" MeasureBase="Nm" Measure="Nm" />
  <Result Name="out_dxMbmax" Type="double" Key="7" Value="= 515.0" />
  <Result Name="@_2638" Type="void" Key="7" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dsigmabmax" Type="double" Key="8" Value="0.2" />
  <Result Name="@_2639" Type="void" Key="8" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dxsigmabmax" Type="double" Key="9" Value="= 350.0" />
  <Result Name="@_2640" Type="void" Key="9" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dtaumax" Type="double" Key="10" Value="0.0" />
  <Result Name="@_2641" Type="void" Key="10" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dxtaumax" Type="double" Key="11" Value="= 0.0" />
  <Result Name="@_2642" Type="void" Key="11" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dsigmazmax" Type="double" Key="12" Value="0.0" />
  <Result Name="@_2643" Type="void" Key="12" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dxsigmazmax" Type="double" Key="13" Value="= 0.0" />
  <Result Name="@_2644" Type="void" Key="13" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dSdmin" Type="double" Key="0" Value="434.7" />
  <Result Name="out_dxSdmin" Type="double" Key="16" Value="= 350.0" />
  <Result Name="@_2647" Type="void" Key="16" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dSfmin" Type="double" Key="0" Value="1738.9" />
  <Result Name="out_dxSfmin" Type="double" Key="17" Value="= 350.0" />
```

```
  <Result Name="@_2648" Type="void" Key="17" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dwmax" Type="double" Key="18" Value="0.002028" />
  <Result Name="@_2649" Type="void" Key="18" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dxwmax" Type="double" Key="19" Value="= 500.0" />
  <Result Name="@_2650" Type="void" Key="19" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dtetamax" Type="double" Key="20" Value="0.000426" />
  <Result Name="@_2651" Type="void" Key="20" MeasureBase="°" Measure="°" />
  <Result Name="out_dxtetamax" Type="double" Key="21" Value="= 0.0" />
  <Result Name="@_2652" Type="void" Key="21" MeasureBase="mm" Measure="mm" />
  <Result Name="out_ddmax" Type="double" Key="45" Value="80.0" />
  <Result Name="@_2721" Type="void" Key="45" MeasureBase="mm" Measure="mm" />
  <Result Name="@_2723" Type="double" Key="46" Value="100.0" />
  <Result Name="@_2724" Type="void" Key="46" MeasureBase="mm" Measure="mm" />
  <Result Name="@_1748" Type="string" Key="0" Value="–" />
  <Result Name="@_1749" Type="string" Key="0" Value="–" />
  <Result Name="out_Rm" Type="double" Key="22" Value="340.0" />
  <Result Name="@_2653" Type="void" Key="22" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_Re" Type="double" Key="23" Value="205.0" />
  <Result Name="@_2654" Type="void" Key="23" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dsigmabf" Type="double" Key="28" Value="287.0" />
  <Result Name="@_2659" Type="void" Key="28" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dtautf" Type="double" Key="29" Value="118.9" />
  <Result Name="@_2660" Type="void" Key="29" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dsigmazzul" Type="double" Key="30" Value="205.0" />
  <Result Name="@_2661" Type="void" Key="30" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dsigmabzul" Type="double" Key="31" Value="170.0" />
  <Result Name="@_2662" Type="void" Key="31" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dtautzul" Type="double" Key="32" Value="102.0" />
  <Result Name="@_2663" Type="void" Key="32" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="@_2988" Type="double" Key="48" Value="500" />
  <Result Name="@_2989" Type="void" Key="48" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dXMb" Type="double" Key="33" Value="5.0" />
  <Result Name="@_2664" Type="void" Key="33" MeasureBase="Nm" Measure="Nm" />
  <Result Name="out_dXsigb" Type="double" Key="34" Value="0.1" />
  <Result Name="@_2665" Type="void" Key="34" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dXtaut" Type="double" Key="35" Value="0.0" />
  <Result Name="@_2666" Type="void" Key="35" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dXsigz" Type="double" Key="36" Value="0.0" />
  <Result Name="@_2667" Type="void" Key="36" MeasureBase="N/mm²" Measure="N/mm²" />
  <Result Name="out_dXSd" Type="double" Key="0" Value="1274.2" />
  <Result Name="out_dXSf" Type="double" Key="0" Value="2885.2" />
  <Result Name="out_dXw" Type="double" Key="38" Value="0.002028" />
  <Result Name="@_2669" Type="void" Key="38" MeasureBase="mm" Measure="mm" />
  <Result Name="out_dXTeta" Type="double" Key="39" Value="0.000001" />
  <Result Name="@_2670" Type="void" Key="39" MeasureBase="°" Measure="°" />
- <Result Name="GsS" Type="list" Key="0" Rows="6" Columns="10">
  <Column Name="GsS_i" Type="integer" Value="1,2,3,4,5,6" />
  <Column Name="GsS_x" Type="double" MeasureBase="mm" Measure="mm"
Value="150,350,450,600,700,900" />
  <Column Name="GsS_da" Type="double" MeasureBase="mm" Measure="mm"
Value="50.5,60,70,80,70,60" />
  <Column Name="GsS_di" Type="double" MeasureBase="mm" Measure="mm" Value="0,0,0,0,0,0"
/>
  <Column Name="GsS_r" Type="double" MeasureBase="mm" Measure="mm" Value="3,3,2,2,3,3"
/>
  <Column Name="GsS_bkzd" Type="double"
Value="10,1.960805,2.164682,2.164682,1.960805,1.897290" />
  <Column Name="GsS_bkb" Type="double"
Value="1.752587,1.813077,2.007916,2.007916,1.813077,1.752587" />
  <Column Name="GsS_bkt" Type="double"
Value="1.371926,1.397504,1.485512,1.485512,1.397504,1.371926" />
  <Column Name="GsS_Sd" Type="double"
Value="640.0223,434.6525,453.6359,474.7353,444.0587,711.3562" />
  <Column Name="GsS_Sf" Type="double"
Value="2419.160,1738.871,2147.649,2247.540,1844.257,2791.338" />
  </Result>
- <Result Name="EF" Type="list" Key="0" Rows="6" Columns="10">
  <Column Name="EF_i" Type="integer" Value="1,2,3,4,5,6" />
  <Column Name="EF_x" Type="double" MeasureBase="mm" Measure="mm"
Value="150,350,450,600,700,900" />
  <Column Name="EF_etak" Type="double" Value="0.8569,0.8569,0.7997,0.7997,0.8569,0.8569"
/>
  <Column Name="EF_b1s" Type="double" Value="0.9646,0.9646,0.9291,0.9291,0.9291,0.9291"
/>
  <Column Name="EF_b1t" Type="double" Value="0.9796,0.9796,0.9592,0.9592,0.9592,0.9592"
/>
```

```
  <Column Name="EF_b2s" Type="double" Value="0.8115,0.7932,0.7706,0.7706,0.7932,0.8115"
/>
  <Column Name="EF_b2t" Type="double" Value="0.8241,0.8075,0.7886,0.7886,0.8075,0.8241"
/>
  <Column Name="EF_akzd" Type="double"
Value="2.047074,2.121191,2.456312,2.456312,2.121191,2.047074" />
  <Column Name="EF_akb" Type="double"
Value="1.878216,1.948803,2.260293,2.260293,1.948803,1.878216" />
  <Column Name="EF_akt" Type="double"
Value="1.434011,1.463860,1.607082,1.607082,1.463860,1.434011" />
  </Result>
- <Result Name="ZlW" Type="list" Key="0" Rows="7" Columns="5">
  <Column Name="ZlW_1" Type="integer" Value="1,2,3,4,5,6,7" />
  <Column Name="ZlW_l" Type="double" MeasureBase="mm" Measure="mm"
Value="150,200,100,150,100,200,150" />
  <Column Name="ZlW_m" Type="double" MeasureBase="kg" Measure="kg"
Value="2.358487,4.439070,3.021034,5.918760,3.021034,4.439070,2.358487" />
  <Column Name="ZlW_I" Type="double" MeasureBase="mm4" Measure="mm4"
Value="6.385e+005,1.272e+006,2.357e+006,4.021e+006,2.357e+006,1.272e+006,6.385e+005" />
  <Column Name="ZlW_J" Type="double" MeasureBase="kgm²" Measure="kgm²"
Value="0.0007518,0.001998,0.00185,0.004735,0.00185,0.001998,0.0007518" />
  </Result>
  </Data>
  </Document>
```

Anexo 2

Código fonte do aplicativo.

```
'============================================================================
'   WARNING!!  This file is overwritten by the UIStyler each time the Styler
'   file is saved.
'
'       Filename:  Scpm.vb
'
'        This file was generated by the NX User Interface Styler
'        Created by: lima_jj
'        Version: NX 5
'              Date: 10-05-2010
'              Time: 10:01
'
'   This template file is overwritten each time the UIStyler dialog is
'   saved.  Any modifications to this file will be lost.
'============================================================================


'============================================================================
'  Purpose:  This TEMPLATE file contains VB.NET source to guide you in the
'  construction of your NX Open application dialog. The generation of your
'  dialog file (.dlg extension) is the first step towards dialog construction
'  within NX.  You must now create a NX Open application that
'  utilizes this file (.dlg).


'  The information in this file provides you with the following:
'
'  1.  Help on how to load and display your UIStyler dialog in NX
'      using APIs provided in NXOpen.UIStyler namespace
'  2.  The empty callback methods (stubs) associated with your dialog items
'      have also been placed in this file. These empty methods have been
'      created simply to start you along with your coding requirements.
'      The method name, argument list and possible return values have already
'      been provided for you.
'============================================================================


'----------------------------------------------------------------------------
' These imports are needed for the following template code
'----------------------------------------------------------------------------
Option Strict Off
Imports System
Imports System.Xml          'Funções XML
Imports System.Text
Imports System.Windows.Forms 'função do Windows
Imports NXOpen
Imports NXOpen.UIStyler


'----------------------------------------------------------------------------
' Represents UI Styler application class
'----------------------------------------------------------------------------
Public Class Scpm
    ' class members
    Private Shared theSession As Session
    Private Shared theUI As UI
    Public Shared theScpm As Scpm
    Private theDialog As Dialog
    Private changeDialog As NXOpen.UIStyler.DialogItem
    Private changeCommonbutton As NXOpen.UIStyler.PushButton

#Region " UI Styler Dialog Designer generator code "
    '----------------------------------------------------------------------------
    ' Constructor for NX Styler class
    '----------------------------------------------------------------------------
    Public Sub New()
        Try
            theSession = Session.GetSession()
            theUI = UI.GetUI()
            theDialog = theUI.Styler.CreateStylerDialog("scpm.dlg")
```

```
            InitializeUIStylerDialog()
        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
    End Sub

    '------------------------------------------------------------------------------
    ' This method is required for UI styler dialog creation
    '------------------------------------------------------------------------------
    Private Sub InitializeUIStylerDialog()
        '-----------------------------------------------------------------------------
        ' The following code snippets initializes all the styler items and associate
        ' respective callbacks. Attributes of the styler item can be accessed and
        ' modified only after calling Show() or RegisterWithUiMenu().
        '-----------------------------------------------------------------------------
        Try
            changeDialog = theDialog.GetStylerItem("UF_STYLER_DIALOG_INDEX",
NXOpen.UIStyler.Dialog.ItemType.DialogItem)
            changeDialog.AddOkayHandler(AddressOf Ok_cb, False)
            changeDialog.AddApplyHandler(AddressOf apply_cb, False)
            changeDialog.AddCancelHandler(AddressOf Cancel_cb, False)
            changeCommonbutton = theDialog.GetStylerItem("COMMONBUTTON",
NXOpen.UIStyler.Dialog.ItemType.PushButton)
            changeCommonbutton.AddActivateHandler(AddressOf COMMONBUTTON_act_cb, False)
        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
    End Sub

#End Region

    Public Shared Function Startup() As Integer
        Try
            theScpm = New Scpm()
            ' The following method registers the dialog with a menu item
            theScpm.RegisterWithUiMenu()

            Startup = 0

        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
    End Function ' Startup ends

    Public Sub RegisterWithUiMenu()
        Try
            Dim isTopDialog As Boolean = False
            theDialog.RegisterWithUiMenu(isTopDialog)
        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
    End Sub

    '------------------------------------------------------------------------------
    '------------------------ UIStyler Callback Functions -------------------------
    Public Function Ok_cb(ByVal eventObject As NXOpen.UIStyler.StylerEvent) As
NXOpen.UIStyler.DialogState
        Try
            '------------------------------------------------------------------------------
------------------
            'Inicio da manipulação dos dados XML
            '------------------------------------------------------------------------------
------------------
            Dim openFileDialog1 As New OpenFileDialog()
            Dim reader As XmlTextReader
            Dim tipo As Integer
            Dim numcol As Integer
            Dim geometria(7, 50) As String
```

```
            Dim geo_temp(7, 50) As String
            Dim modulo As String
            openFileDialog1.InitialDirectory = "d:/"
            openFileDialog1.Filter = "xml files(*.xml)|*.xml| All Files(*.*)|*.*"
            openFileDialog1.FilterIndex = 2
            openFileDialog1.RestoreDirectory = True
            If openFileDialog1.ShowDialog() = System.Windows.Forms.DialogResult.OK Then
                theUI.NXMessageBox.Show("Arquivo", NXMessageBox.DialogType.Information,
openFileDialog1.FileName)
                reader = New XmlTextReader(openFileDialog1.FileName)

                Do While (reader.Read())
                    Select Case reader.NodeType
                        Case XmlNodeType.Element 'Exibir o início do elemento.
                            '====================================
                            'Identifica nome do Modulo de calculo
                            '====================================
                            If (reader.Name = "Document") Then
                                If reader.HasAttributes Then 'Se existirem atributos
                                    While reader.MoveToNextAttribute()
                                        'filtra atributo Rows
                                        If (reader.Name = "CalcName") Then
                                            theUI.NXMessageBox.Show("O Módulo é: ",
NXMessageBox.DialogType.Information, reader.Value)
                                            modulo = reader.Value
                                        End If

                                    End While
                                End If
                            End If
                            '=================================================
                            'se modulo for shaft = filtra informações geometricas
                            '=================================================
                            If (modulo = "Shaft") Then

                                If (reader.Name = "Column") Then
                                    If reader.HasAttributes Then 'Se existirem atributos
                                        While reader.MoveToNextAttribute()
                                            'filtra atributo Rows
                                            If (reader.Name = "Name") Then
                                                'define tag tipo geometria
                                                If reader.Value = "GEO_l" Then tipo = 0
                                                If reader.Value = "GEO_Da_l" Then tipo =
1
                                                If reader.Value = "GEO_Di_l" Then tipo =
2
                                                If reader.Value = "GEO_Da_r" Then tipo =
3
                                                If reader.Value = "GEO_Di_r" Then tipo =
4
                                                If reader.Value = "GEO_r" Then tipo = 5
                                                If reader.Value = "GEO_Rz" Then tipo = 6
                                                If reader.Value = "Beta_kzd" Then tipo =
7
                                            End If
                                            If (reader.Name = "Value") Then
                                                'Cria Matrix geometria

                                                Dim arryRows() As String
                                                arryRows = Split(reader.Value, ",")
                                                For i = 0 To UBound(arryRows)
                                                    geometria(tipo, i) = arryRows(i)
                                                    numcol = i + 1
                                                Next
                                            End If
                                        End While
                                    End If
                                End If

                            End If
                            '=================================================
                            'se modulo for shaft calculation base = filtra informações
geometricas
                            '=================================================
                            If (modulo = "Shaft calculation base") Then
```

```
                                'theUI.NXMessageBox.Show("O valor é: ",
NXMessageBox.DialogType.Information, reader.Value)
                                If (reader.Name = "Column") Then
                                    If reader.HasAttributes Then 'Se existirem atributos
                                        While reader.MoveToNextAttribute()
                                            'Dim couter As Integer =
reader.AttributeCount
                                            ' theUI.NXMessageBox.Show("valor é",
NXMessageBox.DialogType.Information, couter)
                                            'filtra atributo Rows
                                            If (reader.Name = "Name") Then
                                                'define tag tipo geometria
                                                If reader.Value = "DAL_STUFE" Then tipo
= 1
                                                If reader.Value = "DIL_STUFE" Then tipo
= 2
                                                If reader.Value = "DAR_STUFE" Then tipo
= 3
                                                If reader.Value = "DIR_STUFE" Then tipo
= 4
                                                If reader.Value = "L_STUFE" Then tipo =
0
                                                If reader.Value = "R_RZ" Then tipo = 6
                                                If reader.Value = "R" Then tipo = 5
                                                If reader.Value = "D_KERB" Then tipo = 7
                                                'End If
                                            End If
                                            If (reader.Name = "Value") Then
                                                'Cria Matrix geometria
                                                Dim arryRows() As String
                                                arryRows = Split(reader.Value, ",")
                                                For i = 0 To UBound(arryRows)
                                                    geometria(tipo, i) = arryRows(i)
                                                    numcol = 5
                                                Next
                                            End If

                                        End While
                                    End If
                                End If
                            End If


'=====================================================================
                            'se modulo for diferente exibe menssagem de modulo em
desenvolvimento

'=====================================================================
                            If (modulo <> "Shaft") And (modulo <> "Shaft calculation
base") Then
                                CriarSecao(numcol, geometria, tipo, modulo)
                            End If

                        Case XmlNodeType.Text 'Exibir o texto em cada elemento.
                        Case XmlNodeType.EndElement 'Exibir o fim do elemento.
                    End Select
                Loop
                '=====================================================
                'Chamada da função de Criação revisada dia 30/05/2010
                '=====================================================
                CriarSecao(numcol, geometria, tipo, modulo)
                CriarSketchrebaixo(numcol, geometria, tipo, modulo)
                CriarSketchChaveta(numcol, geometria, tipo, modulo)
                CriarChaveta(geometria, tipo)
                Criarrebaixo(geometria, tipo)
            End If

            ' ---- Enter your callback code here -----


        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
        ' Callback acknowledged, terminate dialog
```

```
        ' It is STRONGLY recommanded that you exit your
        ' callback with NXOpen.UIStyler.DialogState.ExitDialog in a ok callback.
        Ok_cb = NXOpen.UIStyler.DialogState.ExitDialog
    End Function

    '------------------------------------------------------------------------------
    ' Callback Name: apply_cb
    ' Following callback is associated with the "changeDialog" Styler item.
    ' Input: eventObject - object of UIStyler.StylerEvent class
    '------------------------------------------------------------------------------
    Public Function apply_cb(ByVal eventObject As NXOpen.UIStyler.StylerEvent) As
NXOpen.UIStyler.DialogState
        Try


            ' ---- Enter your callback code here -----


        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
        ' Callback acknowledged, do not terminate dialog
        ' A return value of NXOpen.UIStyler.DialogState.ExitDialog will not be accepted
        ' for this callback type. You must respond to your apply button.
        apply_cb = NXOpen.UIStyler.DialogState.ContinueDialog
    End Function

    '------------------------------------------------------------------------------
    ' Callback Name: Cancel_cb
    ' Following callback is associated with the "changeDialog" Styler item.
    ' Input: eventObject - object of UIStyler.StylerEvent class
    '------------------------------------------------------------------------------
    Public Function Cancel_cb(ByVal eventObject As NXOpen.UIStyler.StylerEvent) As
NXOpen.UIStyler.DialogState
        Try


            ' ---- Enter your callback code here -----
            theUI.NXMessageBox.Show("Fechando o Aplicativo",
NXMessageBox.DialogType.Information, "Fechar")


        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
        ' Callback acknowledged, terminate dialog
        ' It is STRONGLY recommanded that you exit your
        ' callback with NXOpen.UIStyler.DialogState.ExitDialog in a cancel callback
        ' rather than NXOpen.UIStyler.DialogState.ContinueDialog
        Cancel_cb = NXOpen.UIStyler.DialogState.ExitDialog
    End Function

    '------------------------------------------------------------------------------
    ' Callback Name: COMMONBUTTON_act_cb
    ' Following callback is associated with the "changeCommonbutton" Styler item.
    ' Input: eventObject - object of UIStyler.StylerEvent class
    '------------------------------------------------------------------------------
    Public Function COMMONBUTTON_act_cb(ByVal eventObject As
NXOpen.UIStyler.StylerEvent) As NXOpen.UIStyler.DialogState
        Try


            ' ---- Enter your callback code here -----
            theUI.NXMessageBox.Show("Aviso de Funcionamento",
NXMessageBox.DialogType.Information, "Para Iniciar o Programa e necessário utilizar o
comando OK")

        Catch ex As NXException
            ' ---- Enter your exception handling code here -----
            theUI.NXMessageBox.Show("UI Styler", NXMessageBox.DialogType.Error,
ex.Message)
        End Try
```

```
        ' Callback acknowledged, do not terminate dialog
        COMMONBUTTON_act_cb = NXOpen.UIStyler.DialogState.ContinueDialog
        ' or Callback acknowledged, terminate dialog.
        ' COMMONBUTTON_act_cb = NXOpen.UIStyler.DialogState.ExitDialog
    End Function
    '--------------------------------
    'Função de geração dos módulos
    '--------------------------------

    Public Function CriarSecao(ByVal numcol As Integer, ByVal geometria As Array, ByVal
tipo As Integer, ByVal modulo As String) As Boolean


        If (modulo = "Shaft") Or (modulo = "Shaft calculation base") Then
            'theUI.NXMessageBox.Show("O Módulo " + modulo,
NXMessageBox.DialogType.Information, "entou no criador de seção")


            If (tipo = 7) Then
                Dim theSession As Session = Session.GetSession()
                Dim workPart As Part = theSession.Parts.Work

                Dim displayPart As Part = theSession.Parts.Display


                theUI.NXMessageBox.Show("O número de seções é: ",
NXMessageBox.DialogType.Information, numcol)

                ' ----------------------------------------------
                '   Menu: Insert->Design Feature->Cylinder...
                ' ----------------------------------------------
                For i = 0 To (numcol - 1)

                    Dim nullFeatures_Feature As Features.Feature = Nothing
                    Dim cylinderBuilder As Features.CylinderBuilder
                    cylinderBuilder =
workPart.Features.CreateCylinderBuilder(nullFeatures_Feature)
                    cylinderBuilder.BooleanOption.Type =
GeometricUtilities.BooleanOperation.BooleanType.Create

                    Dim targetBodies(0) As Body
                    Dim nullBody As Body = Nothing

                    targetBodies(0) = nullBody

                    cylinderBuilder.BooleanOption.SetTargetBodies(targetBodies)

                    'cylinderBuilder.Type =
Features.CylinderBuilder.Types.AxisDiameterAndHeight

                    Dim nXObject As NXObject

                    If (i = 0) Then
                        cylinderBuilder.Diameter.RightHandSide = geometria(1, i)

                        cylinderBuilder.Height.RightHandSide = geometria(0, i)
                        nXObject = cylinderBuilder.Commit()
                        Dim expression As Expression
                        expression = cylinderBuilder.Height
                        expression = cylinderBuilder.Diameter
                    End If

                    If (i <> 0) Then

                        Dim edge As Edge = CType((CType(nXObject,
Features.Cylinder)).FindObject("EDGE * 3 * 1"), Edge)

                        'Dim point As Point
                        'point = workPart.Points.CreatePoint(edge,
SmartObject.UpdateOption.WithinModeling)
                        Dim direction As Direction
                        direction = workPart.Directions.CreateDirection(edge,
Sense.Forward, SmartObject.UpdateOption.WithinModeling)
                        'axis.Point = point
                        Dim axis As Axis
                        axis = cylinderBuilder.Axis
```

```
                        Dim nullPoint As Point = Nothing

                        axis.Point = nullPoint

                        axis.Direction = direction

                        cylinderBuilder.Diameter.RightHandSide = geometria(1, i)

                        cylinderBuilder.Height.RightHandSide = geometria(0, i)

                        nXObject = cylinderBuilder.Commit()
                        Dim expression As Expression
                        expression = cylinderBuilder.Height
                        expression = cylinderBuilder.Diameter

                    End If
                    cylinderBuilder.Destroy()
                    workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

                Next
                ' ----------------------------------------------
                '   Menu: View->Operation->Fit
                ' ----------------------------------------------
                workPart.ModelingViews.WorkView.Fit()

                If numcol > 1 Then
                    ' ----------------------------------------------
                    '   Menu: Insert->Combine Bodies->Unite...
                    ' ----------------------------------------------

                    For i = 2 To numcol
                        Dim nullFeatures_BooleanFeature As Features.BooleanFeature =
Nothing
                        Dim booleanBuilder As Features.BooleanBuilder
                        booleanBuilder =
workPart.Features.CreateBooleanBuilder(nullFeatures_BooleanFeature)
                        booleanBuilder.Tolerance = 0.0254
                        booleanBuilder.Operation = Features.Feature.BooleanType.Unite
                        Dim nXObject1 As NXObject
                        Dim body1 As Body =
CType(workPart.Bodies.FindObject("CYLINDER(1)"), Body)
                        Dim added1 As Boolean
                        added1 = booleanBuilder.Targets.Add(body1)
                        'theUI.NXMessageBox.Show("O Objeto é ",
NXMessageBox.DialogType.Information, "CYLINDER(1)")

                        Dim cylinder As String = "CYLINDER( )"
                        cylinder = cylinder.Replace(" ", i)
                        Dim body As Body = CType(workPart.Bodies.FindObject(cylinder),
Body)
                        Dim added As Boolean
                        added = booleanBuilder.Tools.Add(body)
                        'theUI.NXMessageBox.Show("O Objeto é ",
NXMessageBox.DialogType.Information, cylinder)
                        nXObject1 = booleanBuilder.Commit()
                        booleanBuilder.Destroy()
                    Next
                    ' ----------------------------------------------
                    '   Menu: Insert->Detail Feature->Edge Blend...
                    ' ----------------------------------------------
                    For i = 1 To (numcol - 1)

                        Dim nullFeatures_Feature As Features.Feature = Nothing
                        Dim edgeBlendBuilder1 As Features.EdgeBlendBuilder
                        edgeBlendBuilder1 =
workPart.Features.CreateEdgeBlendBuilder(nullFeatures_Feature)
                        Dim blendLimitsData1 As GeometricUtilities.BlendLimitsData
                        blendLimitsData1 = edgeBlendBuilder1.LimitsListData
                        Dim scCollector1 As ScCollector
                        scCollector1 = workPart.ScCollectors.CreateCollector()
                        Dim seedEdges1(0) As Edge
                        Dim cylinder As String
                        Dim cylinder1 As Features.Cylinder
                        Dim edge As String
                        Dim edge1 As Edge
```

```
                        If geometria(1, i - 1) < geometria(1, i) Then

                            cylinder = "CYLINDER( )"
                            cylinder = cylinder.Replace(" ", i)
                            cylinder1 = CType(workPart.Features.FindObject(cylinder),
Features.Cylinder)

                            edge = "EDGE * 3 CYLINDER(_) 2"
                            edge = edge.Replace("_", i + 1)
                            edge1 = CType(cylinder1.FindObject(edge), Edge)
                            seedEdges1(0) = edge1

                        End If

                        If geometria(1, i - 1) > geometria(1, i) Then

                            cylinder = "CYLINDER( )"
                            cylinder = cylinder.Replace(" ", i + 1)
                            cylinder1 = CType(workPart.Features.FindObject(cylinder),
Features.Cylinder)

                            edge = "EDGE * 3 CYLINDER(_) 1"
                            edge = edge.Replace("_", i)
                            edge1 = CType(cylinder1.FindObject(edge), Edge)
                            seedEdges1(0) = edge1

                        End If

                        Dim edgeMultipleSeedTangentRule1 As EdgeMultipleSeedTangentRule
                        edgeMultipleSeedTangentRule1 =
workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges1, 0.5, True)

                        Dim rules1(0) As SelectionIntentRule
                        rules1(0) = edgeMultipleSeedTangentRule1
                        scCollector1.ReplaceRules(rules1, False)

                        edgeBlendBuilder1.Tolerance = 0.0254

                        edgeBlendBuilder1.AllInstancesOption = False

                        edgeBlendBuilder1.RemoveSelfIntersection = True

                        edgeBlendBuilder1.ConvexConcaveY = False

                        edgeBlendBuilder1.RollOverSmoothEdge = True

                        edgeBlendBuilder1.RollOntoEdge = True

                        edgeBlendBuilder1.MoveSharpEdge = True

                        edgeBlendBuilder1.TrimmingOption = False

                        edgeBlendBuilder1.OverlapOption =
Features.EdgeBlendBuilder.Overlap.AnyConvexityRollOver

                        edgeBlendBuilder1.BlendOrder =
Features.EdgeBlendBuilder.OrderOfBlending.ConvexFirst

                        edgeBlendBuilder1.SetbackOption =
Features.EdgeBlendBuilder.Setback.SeparateFromCorner

                        Dim csIndex1 As Integer
                        csIndex1 = edgeBlendBuilder1.AddChainset(scCollector1,
geometria(5, i - 1))

                        Dim feature1 As Features.Feature
                        feature1 = edgeBlendBuilder1.CommitFeature()

                        edgeBlendBuilder1.Destroy()
                    Next
                    theUI.NXMessageBox.Show("Fim", NXMessageBox.DialogType.Information,
"O modelo foi construido")

                End If
            End If
        End If
```

```
        If (modulo <> "Shaft") And (modulo <> "Shaft calculation base") Then
            theUI.NXMessageBox.Show("O Módulo " + modulo,
NXMessageBox.DialogType.Information, "Está em Construção")

        End If

    End Function

    Public Function CriarSketchrebaixo(ByVal numcol As Integer, ByVal geometria As
Array, ByVal tipo As Integer, ByVal modulo As String) As Boolean
        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' ----------------------------------------------
        '   Menu: Insert->Sketch...
        ' ----------------------------------------------

        theSession.BeginTaskEnvironment()

        Dim nullSketch As Sketch = Nothing

        Dim sketchInPlaceBuilder1 As SketchInPlaceBuilder
        sketchInPlaceBuilder1 = workPart.Sketches.CreateSketchInPlaceBuilder(nullSketch)

        Dim section1 As Section
        section1 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim section2 As Section
        section2 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim sketchAlongPathBuilder1 As SketchAlongPathBuilder
        sketchAlongPathBuilder1 =
workPart.Sketches.CreateSketchAlongPathBuilder(nullSketch)

        Dim datumPlane1 As DatumPlane = CType(workPart.Datums.FindObject("DATUM_CSYS(0)
XY plane"), DatumPlane)

        Dim point1 As Point3d = New Point3d(31.205966463595, 0.0, 29.9028996865204)
        sketchInPlaceBuilder1.PlaneOrFace.SetValue(datumPlane1,
workPart.ModelingViews.WorkView, point1)

        sketchInPlaceBuilder1.Axis.Value = Nothing

        Dim taggedObject1 As TaggedObject
        taggedObject1 = sketchInPlaceBuilder1.Axis.Value

        Dim datumAxis1 As DatumAxis = CType(workPart.Datums.FindObject("DATUM_CSYS(0) X
axis"), DatumAxis)

        sketchInPlaceBuilder1.Axis.Value = datumAxis1

        sketchInPlaceBuilder1.ReversePlaneNormal = True

        sketchInPlaceBuilder1.ReverseAxis = False

        Dim nXObject1 As NXObject
        nXObject1 = sketchInPlaceBuilder1.Commit()

        Dim sketch1 As Sketch = CType(nXObject1, Sketch)

        Dim feature1 As Features.Feature
        feature1 = sketch1.Feature

        sketch1.Activate(Sketch.ViewReorient.True)

        sketchInPlaceBuilder1.Destroy()

        sketchAlongPathBuilder1.Destroy()

        section2.Destroy()

        section1.Destroy()

        theSession.ActiveSketch.SetName("SKETCH_000")
```

```
        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

        ' ----------------------------------------------
        ' Creating rectangle using By 2 Points method
        ' ----------------------------------------------
        Dim startPoint1 As Point3d = New Point3d(10.0, 0.0, 55.0)
        Dim endPoint1 As Point3d = New Point3d(30.0, 0.0, 55.0)
        Dim line1 As Line
        line1 = workPart.Curves.CreateLine(startPoint1, endPoint1)

        Dim startPoint2 As Point3d = New Point3d(30.0, 0.0, 55.0)
        Dim endPoint2 As Point3d = New Point3d(30.0, 0.0, 50.0)
        Dim line2 As Line
        line2 = workPart.Curves.CreateLine(startPoint2, endPoint2)

        Dim startPoint3 As Point3d = New Point3d(30.0, 0.0, 50.0)
        Dim endPoint3 As Point3d = New Point3d(10.0, 0.0, 50.0)
        Dim line3 As Line
        line3 = workPart.Curves.CreateLine(startPoint3, endPoint3)

        Dim startPoint4 As Point3d = New Point3d(10.0, 0.0, 50.0)
        Dim endPoint4 As Point3d = New Point3d(10.0, 0.0, 55.0)
        Dim line4 As Line
        line4 = workPart.Curves.CreateLine(startPoint4, endPoint4)

        theSession.ActiveSketch.AddGeometry(line1)

        theSession.ActiveSketch.AddGeometry(line2)

        theSession.ActiveSketch.AddGeometry(line3)

        theSession.ActiveSketch.AddGeometry(line4)

        Dim geom1 As Sketch.ConstraintGeometry
        geom1.Geometry = line1
        geom1.PointType = Sketch.ConstraintPointType.None
        geom1.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint1 As SketchGeometricConstraint
        sketchGeometricConstraint1 =
theSession.ActiveSketch.CreateHorizontalConstraint(geom1)

        Dim conGeom1_1 As Sketch.ConstraintGeometry
        conGeom1_1.Geometry = line1
        conGeom1_1.PointType = Sketch.ConstraintPointType.None
        conGeom1_1.SplineDefiningPointIndex = 0
        Dim conGeom2_1 As Sketch.ConstraintGeometry
        conGeom2_1.Geometry = line2
        conGeom2_1.PointType = Sketch.ConstraintPointType.None
        conGeom2_1.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint2 As SketchGeometricConstraint
        sketchGeometricConstraint2 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_1, conGeom2_1)

        Dim conGeom1_2 As Sketch.ConstraintGeometry
        conGeom1_2.Geometry = line2
        conGeom1_2.PointType = Sketch.ConstraintPointType.None
        conGeom1_2.SplineDefiningPointIndex = 0
        Dim conGeom2_2 As Sketch.ConstraintGeometry
        conGeom2_2.Geometry = line3
        conGeom2_2.PointType = Sketch.ConstraintPointType.None
        conGeom2_2.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint3 As SketchGeometricConstraint
        sketchGeometricConstraint3 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_2, conGeom2_2)

        Dim conGeom1_3 As Sketch.ConstraintGeometry
        conGeom1_3.Geometry = line3
        conGeom1_3.PointType = Sketch.ConstraintPointType.None
        conGeom1_3.SplineDefiningPointIndex = 0
        Dim conGeom2_3 As Sketch.ConstraintGeometry
        conGeom2_3.Geometry = line4
        conGeom2_3.PointType = Sketch.ConstraintPointType.None
        conGeom2_3.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint4 As SketchGeometricConstraint
```

```
        sketchGeometricConstraint4 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_3, conGeom2_3)

        Dim conGeom1_4 As Sketch.ConstraintGeometry
        conGeom1_4.Geometry = line4
        conGeom1_4.PointType = Sketch.ConstraintPointType.None
        conGeom1_4.SplineDefiningPointIndex = 0
        Dim conGeom2_4 As Sketch.ConstraintGeometry
        conGeom2_4.Geometry = line1
        conGeom2_4.PointType = Sketch.ConstraintPointType.None
        conGeom2_4.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint5 As SketchGeometricConstraint
        sketchGeometricConstraint5 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_4, conGeom2_4)

        'theSession.ActiveSketch.Update()

        Dim geoms1(3) As SmartObject
        geoms1(0) = line1
        geoms1(1) = line2
        geoms1(2) = line3
        geoms1(3) = line4
        theSession.ActiveSketch.UpdateConstraintDisplay(geoms1)

        Dim geoms2(3) As SmartObject
        geoms2(0) = line1
        geoms2(1) = line2
        geoms2(2) = line3
        geoms2(3) = line4
        theSession.ActiveSketch.UpdateDimensionDisplay(geoms2)

        ' ---------------------------------------------
        '   Menu: Sketch->Finish Sketch
        ' ---------------------------------------------
        theSession.ActiveSketch.Deactivate(Sketch.ViewReorient.True,
Sketch.UpdateLevel.Model)

        theSession.EndTaskEnvironment()


    End Function

    Public Function Criarrebaixo(ByVal geometria As Array, ByVal tipo As Integer) As
Boolean


        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' ---------------------------------------------
        '   Menu: Insert->Design Feature->Revolve...
        ' ---------------------------------------------
        Dim section1 As Section
        section1 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim nullFeatures_Feature As Features.Feature = Nothing

        Dim revolveBuilder1 As Features.RevolveBuilder
        revolveBuilder1 = workPart.Features.CreateRevolveBuilder(nullFeatures_Feature)

        revolveBuilder1.BooleanOperation.Type =
GeometricUtilities.BooleanOperation.BooleanType.Create

        Dim targetBodies1(0) As Body
        Dim nullBody As Body = Nothing

        targetBodies1(0) = nullBody
        revolveBuilder1.BooleanOperation.SetTargetBodies(targetBodies1)

        revolveBuilder1.Tolerance = 0.0254

        revolveBuilder1.Section = section1

        section1.DistanceTolerance = 0.0254
```

```
        section1.ChainingTolerance = 0.02413

        Dim features1(0) As Features.Feature
        Dim sketchFeature1 As Features.SketchFeature =
CType(workPart.Features.FindObject("SKETCH(14)"), Features.SketchFeature)

        features1(0) = sketchFeature1
        Dim curveFeatureRule1 As CurveFeatureRule
        curveFeatureRule1 = workPart.ScRuleFactory.CreateRuleCurveFeature(features1)

        section1.AllowSelfIntersection(False)

        Dim rules1(0) As SelectionIntentRule
        rules1(0) = curveFeatureRule1
        Dim nullNXObject As NXObject = Nothing

        Dim helpPoint1 As Point3d = New Point3d(0.0, 0.0, 0.0)
        section1.AddToSection(rules1, nullNXObject, nullNXObject, nullNXObject,
helpPoint1, Section.Mode.Create, False)

        revolveBuilder1.Section = section1

        Dim origin1 As Point3d = New Point3d(0.0, 0.0, 0.0)
        Dim vector1 As Vector3d = New Vector3d(0.0, 0.0, 1.0)
        Dim direction1 As Direction
        direction1 = workPart.Directions.CreateDirection(origin1, vector1,
SmartObject.UpdateOption.WithinModeling)

        Dim nullPoint As Point = Nothing

        Dim axis1 As Axis
        axis1 = workPart.Axes.CreateAxis(nullPoint, direction1,
SmartObject.UpdateOption.WithinModeling)

        revolveBuilder1.Axis = axis1

        Dim cylinder1 As Features.Cylinder =
CType(workPart.Features.FindObject("CYLINDER(1)"), Features.Cylinder)

        Dim edge1 As Edge = CType(cylinder1.FindObject("EDGE * 2 * 3"), Edge)

        Dim point1 As Point
        point1 = workPart.Points.CreatePoint(edge1,
SmartObject.UpdateOption.WithinModeling)

        axis1.Point = point1

        revolveBuilder1.Axis = axis1

        revolveBuilder1.Limits.EndExtend.Value.RightHandSide = "360"

        revolveBuilder1.ParentFeatureInternal = False

        Dim feature1 As Features.Feature
        feature1 = revolveBuilder1.CommitFeature()

        Dim expression1 As Expression = revolveBuilder1.Limits.StartExtend.Value

        Dim expression2 As Expression = revolveBuilder1.Limits.EndExtend.Value

        revolveBuilder1.Destroy()

        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

        ' ------------------------------------------------
        '   Menu: Insert->Combine Bodies->Subtract...
        ' ------------------------------------------------
        Dim nullFeatures_BooleanFeature As Features.BooleanFeature = Nothing

        Dim booleanBuilder1 As Features.BooleanBuilder
        booleanBuilder1 =
workPart.Features.CreateBooleanBuilder(nullFeatures_BooleanFeature)

        booleanBuilder1.Tolerance = 0.0254
```

```
        booleanBuilder1.Operation = Features.Feature.BooleanType.Subtract

        Dim body1 As Body = CType(workPart.Bodies.FindObject("CYLINDER(1)"), Body)

        Dim added1 As Boolean
        added1 = booleanBuilder1.Targets.Add(body1)

        Dim body2 As Body = CType(workPart.Bodies.FindObject("REVOLVED(17)"), Body)

        Dim added2 As Boolean
        added2 = booleanBuilder1.Tools.Add(body2)

        Dim nXObject1 As NXObject
        nXObject1 = booleanBuilder1.Commit()

        booleanBuilder1.Destroy()

        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

    End Function

    Public Function CriarSketchChaveta(ByVal numcol As Integer, ByVal geometria As
Array, ByVal tipo As Integer, ByVal modulo As String) As Boolean

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' ----------------------------------------------
        '   Menu: Insert->Sketch...
        ' ----------------------------------------------
        Dim markId1 As Session.UndoMarkId
        markId1 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Enter
Sketcher")

        Dim markId2 As Session.UndoMarkId
        markId2 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Update Model
from Sketcher")

        theSession.BeginTaskEnvironment()

        Dim markId3 As Session.UndoMarkId
        markId3 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Start")

        Dim nullSketch As Sketch = Nothing

        Dim sketchInPlaceBuilder1 As SketchInPlaceBuilder
        sketchInPlaceBuilder1 = workPart.Sketches.CreateSketchInPlaceBuilder(nullSketch)

        Dim section1 As Section
        section1 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim section2 As Section
        section2 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim sketchAlongPathBuilder1 As SketchAlongPathBuilder
        sketchAlongPathBuilder1 =
workPart.Sketches.CreateSketchAlongPathBuilder(nullSketch)

        theSession.SetUndoMarkName(markId3, "Create Sketch Dialog")

        Dim datumPlane1 As DatumPlane = CType(workPart.Datums.FindObject("DATUM_CSYS(0)
XZ plane"), DatumPlane)

        Dim point1 As Point3d = New Point3d(-0.00000000000000177635683940025,
30.5245808388164, 30.0177364243762)
        sketchInPlaceBuilder1.PlaneOrFace.SetValue(datumPlane1,
workPart.ModelingViews.WorkView, point1)

        sketchInPlaceBuilder1.Axis.Value = Nothing

        Dim datumAxis1 As DatumAxis = CType(workPart.Datums.FindObject("DATUM_CSYS(0) Y
axis"), DatumAxis)

        sketchInPlaceBuilder1.Axis.Value = datumAxis1
```

```
        sketchInPlaceBuilder1.ReverseAxis = False

        sketchInPlaceBuilder1.PlaneOption = Sketch.PlaneOption.NewPlane

        sketchInPlaceBuilder1.Plane.SetMethod(PlaneTypes.MethodType.Distance)

        Dim geom1(0) As NXObject
        geom1(0) = datumPlane1
        sketchInPlaceBuilder1.Plane.SetGeometry(geom1)

        sketchInPlaceBuilder1.Plane.SetFlip(False)

        sketchInPlaceBuilder1.Plane.SetReverseSide(False)

        Dim expression1 As Expression
        expression1 = sketchInPlaceBuilder1.Plane.Expression

        expression1.RightHandSide = "-60"

        sketchInPlaceBuilder1.Plane.SetAlternate(PlaneTypes.AlternateType.One)

        sketchInPlaceBuilder1.Plane.Evaluate()

        sketchInPlaceBuilder1.Plane.SetMethod(PlaneTypes.MethodType.Distance)

        Dim geom2(0) As NXObject
        geom2(0) = datumPlane1
        sketchInPlaceBuilder1.Plane.SetGeometry(geom2)

        sketchInPlaceBuilder1.Plane.SetFlip(False)

        sketchInPlaceBuilder1.Plane.SetReverseSide(False)

        Dim expression2 As Expression
        expression2 = sketchInPlaceBuilder1.Plane.Expression

        expression2.RightHandSide = "-60"

        sketchInPlaceBuilder1.Plane.SetAlternate(PlaneTypes.AlternateType.One)

        sketchInPlaceBuilder1.Plane.Evaluate()

        Dim cylinder1 As Features.Cylinder =
CType(workPart.Features.FindObject("CYLINDER(1)"), Features.Cylinder)

        Dim face1 As Face = CType(cylinder1.FindObject("FACE 3"), Face)

        Dim line1 As Line
        line1 = workPart.Lines.CreateFaceAxis(face1,
SmartObject.UpdateOption.WithinModeling)

        sketchInPlaceBuilder1.Plane.SetMethod(PlaneTypes.MethodType.Distance)

        Dim geom3(0) As NXObject
        geom3(0) = datumPlane1
        sketchInPlaceBuilder1.Plane.SetGeometry(geom3)

        sketchInPlaceBuilder1.Plane.SetFlip(False)

        sketchInPlaceBuilder1.Plane.SetReverseSide(False)

        Dim expression3 As Expression
        expression3 = sketchInPlaceBuilder1.Plane.Expression

        expression3.RightHandSide = "25"

        sketchInPlaceBuilder1.Plane.SetAlternate(PlaneTypes.AlternateType.One)

        sketchInPlaceBuilder1.Plane.Evaluate()

        Dim markId4 As Session.UndoMarkId
        markId4 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Create
Sketch")

        Dim flip1 As Boolean
```

```
        flip1 = sketchInPlaceBuilder1.Plane.Flip

        Dim nXObject1 As NXObject
        nXObject1 = sketchInPlaceBuilder1.Commit()

        Dim sketch1 As Sketch = CType(nXObject1, Sketch)

        Dim feature1 As Features.Feature
        feature1 = sketch1.Feature

        Dim nErrs1 As Integer
        nErrs1 = theSession.UpdateManager.DoUpdate(markId3)

        sketch1.Activate(Sketch.ViewReorient.True)

        theSession.DeleteUndoMark(markId4, Nothing)

        theSession.SetUndoMarkName(markId3, "Create Sketch")

        sketchInPlaceBuilder1.Destroy()

        sketchAlongPathBuilder1.Destroy()

        section2.Destroy()

        section1.Destroy()

        theSession.DeleteUndoMarksUpToMark(markId2, Nothing, True)

        Dim markId5 As Session.UndoMarkId
        markId5 = theSession.SetUndoMark(Session.MarkVisibility.Visible, "Open Sketch")

        theSession.ActiveSketch.SetName("SKETCH_001")

        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

        Dim markId6 As Session.UndoMarkId
        markId6 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Profile
short list")

        ' ----------------------------------------------
        '   Menu: Pan
        ' ----------------------------------------------
        Dim origin1 As Point3d = New Point3d(25.0, 9.07754337374811, 60.6600219144083)
        workPart.ModelingViews.WorkView.SetOrigin(origin1)

        ' ----------------------------------------------
        '   Menu: Insert->Curve->Rectangle...
        ' ----------------------------------------------
        Dim origin2 As Point3d = New Point3d(25.0, 9.07754337374811, 60.6600219144083)
        workPart.ModelingViews.WorkView.SetOrigin(origin2)

        theSession.DeleteUndoMark(markId6, "Curve")

        Dim markId7 As Session.UndoMarkId
        markId7 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Profile
short list")

        Dim markId8 As Session.UndoMarkId
        markId8 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Create
Rectangle")

        theSession.SetUndoMarkVisibility(markId8, "Create Rectangle",
Session.MarkVisibility.Visible)

        ' ----------------------------------------------
        ' Creating rectangle using By 2 Points method
        ' ----------------------------------------------
        Dim markId9 As Session.UndoMarkId
        markId9 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Create
Rectangle")

        theSession.SetUndoMarkVisibility(markId9, "Create Rectangle",
Session.MarkVisibility.Visible)

        ' ----------------------------------------------
```

```
        ' Creating rectangle using By 2 Points method
        ' ----------------------------------------------
        Dim startPoint1 As Point3d = New Point3d(25.0, -(50 / 8), 187.0)
        Dim endPoint1 As Point3d = New Point3d(25.0, (50 / 8), 187.0)
        Dim line2 As Line
        line2 = workPart.Curves.CreateLine(startPoint1, endPoint1)

        Dim startPoint2 As Point3d = New Point3d(25.0, (50 / 8), 187.0)
        Dim endPoint2 As Point3d = New Point3d(25.0, (50 / 8), 141.0)
        Dim line3 As Line
        line3 = workPart.Curves.CreateLine(startPoint2, endPoint2)

        Dim startPoint3 As Point3d = New Point3d(25.0, (50 / 8), 141.0)
        Dim endPoint3 As Point3d = New Point3d(25.0, -(50 / 8), 141.0)
        Dim line4 As Line
        line4 = workPart.Curves.CreateLine(startPoint3, endPoint3)

        Dim startPoint4 As Point3d = New Point3d(25.0, -(50 / 8), 141.0)
        Dim endPoint4 As Point3d = New Point3d(25.0, -(50 / 8), 187.0)
        Dim line5 As Line
        line5 = workPart.Curves.CreateLine(startPoint4, endPoint4)

        theSession.ActiveSketch.AddGeometry(line2)

        theSession.ActiveSketch.AddGeometry(line3)

        theSession.ActiveSketch.AddGeometry(line4)

        theSession.ActiveSketch.AddGeometry(line5)

        Dim geom4 As Sketch.ConstraintGeometry
        geom4.Geometry = line2
        geom4.PointType = Sketch.ConstraintPointType.None
        geom4.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint1 As SketchGeometricConstraint
        sketchGeometricConstraint1 =
theSession.ActiveSketch.CreateHorizontalConstraint(geom4)

        Dim conGeom1_1 As Sketch.ConstraintGeometry
        conGeom1_1.Geometry = line2
        conGeom1_1.PointType = Sketch.ConstraintPointType.None
        conGeom1_1.SplineDefiningPointIndex = 0
        Dim conGeom2_1 As Sketch.ConstraintGeometry
        conGeom2_1.Geometry = line3
        conGeom2_1.PointType = Sketch.ConstraintPointType.None
        conGeom2_1.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint2 As SketchGeometricConstraint
        sketchGeometricConstraint2 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_1, conGeom2_1)

        Dim conGeom1_2 As Sketch.ConstraintGeometry
        conGeom1_2.Geometry = line3
        conGeom1_2.PointType = Sketch.ConstraintPointType.None
        conGeom1_2.SplineDefiningPointIndex = 0
        Dim conGeom2_2 As Sketch.ConstraintGeometry
        conGeom2_2.Geometry = line4
        conGeom2_2.PointType = Sketch.ConstraintPointType.None
        conGeom2_2.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint3 As SketchGeometricConstraint
        sketchGeometricConstraint3 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_2, conGeom2_2)

        Dim conGeom1_3 As Sketch.ConstraintGeometry
        conGeom1_3.Geometry = line4
        conGeom1_3.PointType = Sketch.ConstraintPointType.None
        conGeom1_3.SplineDefiningPointIndex = 0
        Dim conGeom2_3 As Sketch.ConstraintGeometry
        conGeom2_3.Geometry = line5
        conGeom2_3.PointType = Sketch.ConstraintPointType.None
        conGeom2_3.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint4 As SketchGeometricConstraint
        sketchGeometricConstraint4 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_3, conGeom2_3)

        Dim conGeom1_4 As Sketch.ConstraintGeometry
        conGeom1_4.Geometry = line5
```

```
        conGeom1_4.PointType = Sketch.ConstraintPointType.None
        conGeom1_4.SplineDefiningPointIndex = 0
        Dim conGeom2_4 As Sketch.ConstraintGeometry
        conGeom2_4.Geometry = line2
        conGeom2_4.PointType = Sketch.ConstraintPointType.None
        conGeom2_4.SplineDefiningPointIndex = 0
        Dim sketchGeometricConstraint5 As SketchGeometricConstraint
        sketchGeometricConstraint5 =
theSession.ActiveSketch.CreatePerpendicularConstraint(conGeom1_4, conGeom2_4)



        Dim geoms1(3) As SmartObject
        geoms1(0) = line2
        geoms1(1) = line3
        geoms1(2) = line4
        geoms1(3) = line5
        theSession.ActiveSketch.UpdateConstraintDisplay(geoms1)

        Dim geoms2(3) As SmartObject
        geoms2(0) = line2
        geoms2(1) = line3
        geoms2(2) = line4
        geoms2(3) = line5
        theSession.ActiveSketch.UpdateDimensionDisplay(geoms2)
        Dim nullNXObject As NXObject = Nothing
        ' theSession.ActiveSketch.Update()
        ' -----------------------------------------------
        '   Menu: Sketch->Finish Sketch
        ' -----------------------------------------------
        theSession.ActiveSketch.Deactivate(Sketch.ViewReorient.True,
Sketch.UpdateLevel.Model)

        theSession.DeleteUndoMarksSetInTaskEnvironment()

        theSession.EndTaskEnvironment()


    End Function
    Public Function CriarChaveta(ByVal geometria As Array, ByVal tipo As Integer) As
Boolean

        Dim theSession As Session = Session.GetSession()
        Dim workPart As Part = theSession.Parts.Work

        Dim displayPart As Part = theSession.Parts.Display

        ' -----------------------------------------------
        '   Menu: Insert->Design Feature->Extrude...
        ' -----------------------------------------------
        Dim section1 As Section
        section1 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim nullFeatures_Feature As Features.Feature = Nothing

        Dim extrudeBuilder1 As Features.ExtrudeBuilder
        extrudeBuilder1 = workPart.Features.CreateExtrudeBuilder(nullFeatures_Feature)

        extrudeBuilder1.Section = section1

        extrudeBuilder1.DistanceTolerance = 0.0254

        extrudeBuilder1.Offset.SetStartOffset("0")

        extrudeBuilder1.Offset.SetEndOffset("5")

        extrudeBuilder1.BooleanOperation.Type =
GeometricUtilities.BooleanOperation.BooleanType.Create

        Dim targetBodies1(0) As Body
        Dim nullBody As Body = Nothing

        targetBodies1(0) = nullBody
        extrudeBuilder1.BooleanOperation.SetTargetBodies(targetBodies1)

        section1.DistanceTolerance = 0.0254
```

```
        section1.ChainingTolerance = 0.02413

        Dim features1(0) As Features.Feature
        Dim sketchFeature1 As Features.SketchFeature =
CType(workPart.Features.FindObject("SKETCH(16)"), Features.SketchFeature)

        features1(0) = sketchFeature1
        Dim curveFeatureRule1 As CurveFeatureRule
        curveFeatureRule1 = workPart.ScRuleFactory.CreateRuleCurveFeature(features1)

        section1.AllowSelfIntersection(False)

        Dim rules1(0) As SelectionIntentRule
        rules1(0) = curveFeatureRule1
        Dim nullNXObject As NXObject = Nothing

        Dim helpPoint1 As Point3d = New Point3d(0.0, 0.0, 0.0)
        section1.AddToSection(rules1, nullNXObject, nullNXObject, nullNXObject,
helpPoint1, Section.Mode.Create, False)

        Dim sketch1 As Sketch = CType(workPart.Sketches.FindObject("SKETCH_001"),
Sketch)

        Dim direction1 As Direction
        direction1 = workPart.Directions.CreateDirection(sketch1, Sense.Forward,
SmartObject.UpdateOption.WithinModeling)

        extrudeBuilder1.Direction = direction1

        extrudeBuilder1.Limits.StartExtend.SetValue("0")

        extrudeBuilder1.Limits.EndExtend.SetValue("-50/4")

        extrudeBuilder1.Limits.EndExtend.Value.RightHandSide = "-50/4"

        Dim markId5 As Session.UndoMarkId
        markId5 = theSession.SetUndoMark(Session.MarkVisibility.Invisible, "Extrude")

        extrudeBuilder1.ParentFeatureInternal = False

        Dim feature1 As Features.Feature
        feature1 = extrudeBuilder1.CommitFeature()

        theSession.DeleteUndoMark(markId5, Nothing)

        Dim expression1 As Expression = extrudeBuilder1.Limits.StartExtend.Value

        Dim expression2 As Expression = extrudeBuilder1.Limits.EndExtend.Value

        extrudeBuilder1.Destroy()

        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

        ' ----------------------------------------------
        '   Menu: Insert->Combine Bodies->Subtract...
        ' ----------------------------------------------

        Dim nullFeatures_BooleanFeature As Features.BooleanFeature = Nothing

        Dim booleanBuilder1 As Features.BooleanBuilder
        booleanBuilder1 =
workPart.Features.CreateBooleanBuilder(nullFeatures_BooleanFeature)

        booleanBuilder1.Tolerance = 0.0254

        booleanBuilder1.Operation = Features.Feature.BooleanType.Subtract

        Dim body1 As Body = CType(workPart.Bodies.FindObject("CYLINDER(1)"), Body)

        Dim added1 As Boolean
        added1 = booleanBuilder1.Targets.Add(body1)

        Dim body2 As Body = CType(workPart.Bodies.FindObject("EXTRUDE(17)"), Body)

        Dim added2 As Boolean
```

```
        added2 = booleanBuilder1.Tools.Add(body2)

        Dim nXObject1 As NXObject
        nXObject1 = booleanBuilder1.Commit()

        booleanBuilder1.Destroy()

        workPart.FacetedBodies.DeleteTemporaryFacesAndEdges()

        ' ----------------------------------------------
        '   Menu: Insert->Detail Feature->Edge Blend...
        ' ----------------------------------------------

        Dim edgeBlendBuilder1 As Features.EdgeBlendBuilder
        edgeBlendBuilder1 =
workPart.Features.CreateEdgeBlendBuilder(nullFeatures_Feature)

        Dim blendLimitsData1 As GeometricUtilities.BlendLimitsData
        blendLimitsData1 = edgeBlendBuilder1.LimitsListData

        Dim section2 As Section
        section2 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        Dim section3 As Section
        section3 = workPart.Sections.CreateSection(0.02413, 0.0254, 0.5)

        section2.DistanceTolerance = 0.0254

        section2.ChainingTolerance = 0.02413

        section3.DistanceTolerance = 0.0254

        section3.ChainingTolerance = 0.02413

        Dim scCollector1 As ScCollector
        scCollector1 = workPart.ScCollectors.CreateCollector()

        Dim seedEdges1(0) As Edge
        Dim extrude1 As Features.Extrude = CType(feature1, Features.Extrude)

        Dim edge1 As Edge = CType(extrude1.FindObject("EDGE * 140 * 150"), Edge)

        seedEdges1(0) = edge1
        Dim edgeMultipleSeedTangentRule1 As EdgeMultipleSeedTangentRule
        edgeMultipleSeedTangentRule1 =
workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges1, 0.5, True)

        Dim rules2(0) As SelectionIntentRule
        rules2(0) = edgeMultipleSeedTangentRule1
        scCollector1.ReplaceRules(rules2, False)

        Dim seedEdges2(1) As Edge
        seedEdges2(0) = edge1
        Dim edge2 As Edge = CType(extrude1.FindObject("EDGE * 150 * 160"), Edge)

        seedEdges2(1) = edge2
        Dim edgeMultipleSeedTangentRule2 As EdgeMultipleSeedTangentRule
        edgeMultipleSeedTangentRule2 =
workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges2, 0.5, True)

        Dim rules3(0) As SelectionIntentRule
        rules3(0) = edgeMultipleSeedTangentRule2
        scCollector1.ReplaceRules(rules3, False)

        Dim seedEdges3(2) As Edge
        seedEdges3(0) = edge1
        seedEdges3(1) = edge2
        Dim edge3 As Edge = CType(extrude1.FindObject("EDGE * 160 * 170"), Edge)

        seedEdges3(2) = edge3
        Dim edgeMultipleSeedTangentRule3 As EdgeMultipleSeedTangentRule
        edgeMultipleSeedTangentRule3 =
workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges3, 0.5, True)

        Dim rules4(0) As SelectionIntentRule
        rules4(0) = edgeMultipleSeedTangentRule3
```

```
        scCollector1.ReplaceRules(rules4, False)

        Dim seedEdges4(3) As Edge
        seedEdges4(0) = edge1
        seedEdges4(1) = edge2
        seedEdges4(2) = edge3
        Dim edge4 As Edge = CType(extrude1.FindObject("EDGE * 140 * 170"), Edge)

        seedEdges4(3) = edge4
        Dim edgeMultipleSeedTangentRule4 As EdgeMultipleSeedTangentRule
        edgeMultipleSeedTangentRule4 =
workPart.ScRuleFactory.CreateRuleEdgeMultipleSeedTangent(seedEdges4, 0.5, True)

        Dim rules5(0) As SelectionIntentRule
        rules5(0) = edgeMultipleSeedTangentRule4
        scCollector1.ReplaceRules(rules5, False)

        edgeBlendBuilder1.Tolerance = 0.0254

        edgeBlendBuilder1.AllInstancesOption = False

        edgeBlendBuilder1.RemoveSelfIntersection = True

        edgeBlendBuilder1.ConvexConcaveY = False

        edgeBlendBuilder1.RollOverSmoothEdge = True

        edgeBlendBuilder1.RollOntoEdge = True

        edgeBlendBuilder1.MoveSharpEdge = True

        edgeBlendBuilder1.TrimmingOption = False

        edgeBlendBuilder1.OverlapOption =
Features.EdgeBlendBuilder.Overlap.AnyConvexityRollOver

        edgeBlendBuilder1.BlendOrder =
Features.EdgeBlendBuilder.OrderOfBlending.ConvexFirst

        edgeBlendBuilder1.SetbackOption =
Features.EdgeBlendBuilder.Setback.SeparateFromCorner

        Dim csIndex1 As Integer
        csIndex1 = edgeBlendBuilder1.AddChainset(scCollector1, "6.25")

        Dim feature2 As Features.Feature
        feature2 = edgeBlendBuilder1.CommitFeature()

        edgeBlendBuilder1.Destroy()

        section3.Destroy()

        section2.Destroy()
    End Function
End Class
```