CATIA

Universidade do Porto
Faculdade de Engenharia
FEUP

Faculdade de Engenharia Universidade do Porto

DEMEG

Departamento de Engenharia Mecânica e Gestão Industrial

Projecto de Fim de Curso

Optimização e Personalização de uma Ferramenta CAD/CAM de Acordo com os Requisitos Próprios da Indústria de Moldes

CADTECH

Elaborado por:
Nelson Luis Vieira Pinto

Supervisionado por:
Eng.º Joaquim Oliveira Fonseca - FEUP
Eng.º Pedro Pinho - CADTECH

Julho de 2003

UNIÃO EUROPEIA
Fundo Social Europeu

prodep III
Mais Educação

Agradecimentos

Quero deixar bem expressos os meus mais sinceros agradecimentos a todos aqueles que fizeram com que fosse possível a realização deste Projecto de Fim de Curso, em especial, ao Eng.º Joaquim Oliveira Fonseca da secção de Desenho Industrial da Faculdade de Engenharia da Universidade do Porto, por toda a sua disponibilidade e apoio ao longo da duração deste Projecto, sendo que, o seu profundo conhecimento de tudo o que envolve a Industria de Moldes revelou-se fundamental. Ainda, gostava de agradecer à empresa CadTech Ibérica, S. A., pela oportunidade da realização deste Estágio Curricular, em especial ao meu orientador na empresa, o Eng.º Pedro Pinho, por ter posto ao meu dispor todos os meios necessários à realização deste Projecto; gostava também de deixar uma palavra especial de agradecimento a toda a equipa da CadTech, pelo seu total apoio e disponibilidade.

O Autor

Índice

Introdução

1

1. Introdução

Na definição de um molde influem muitas regras e conhecimentos (alguns expressáveis matematicamente e outros não), que não ficam expressos pela geometria de um modelo CAD. Seria verdadeiramente interessante que um componente, ao ser inserido num conjunto, respeitasse não apenas as dimensões estipuladas pelo utilizador, mas também as regras de construção (distâncias de segurança, dimensões "deduzidas", etc.) e que procurasse as referências de que necessita para obedecer às normas próprias deste tipo de indústria. Pretende-se também, automatizar e reaproveitar ao máximo os conhecimentos que levam a opção dos diferentes componentes nas várias fases do Projecto. Mais ainda, o respeito pelas interdependências que devem ser mantidas aquando de uma modificação de qualquer componente ou conjunto, assim como todas as verificações e regras que afectam a geometria devem ser recalculadas.

O desafio deste trabalho começa na definição de componentes "inteligentes" que se adaptem ao molde e à função que devem cumprir, e termina na introdução de regras lógicas e funcionais (não apenas matemáticas ou dimensionais) que abreviem o mais possível o estudo de alternativas ou modificações geométricas impostas por alterações da peça a moldar.

Numa primeira fase, terá de haver uma troca de ideias e conhecimentos com alguns fabricantes de Moldes, para verificar quais são as necessidades mais comuns no desenvolvimento de um projecto de um Molde, para posterior personalização das ferramentas utilizadas. Para este efeito, foi visitado um dos maiores centros de desenvolvimento da Indústria de Moldes em Portugal, a Marinha Grande.

No princípio do século XX, iniciou-se na Marinha Grande a produção de Moldes para a Indústria do Vidro, até aí importados da Alemanha e da Áustria. Seria esta a base que levaria ao progresso da Indústria de Moldes para Plásticos, em Portugal. Esta Indústria desenvolveu-se com a importação de tecnologia estrangeira e em 1955 iniciou-se a exportação com a venda dos primeiros Moldes à Grã-Bretanha. Em 1980, a Indústria de Moldes nacional já exportava para mais de 50 países e só na área da Marinha Grande, existiam cerca de 64 empresas em laboração, empregando cerca de 2.000 pessoas. Hoje o sector da Indústria de Moldes Português, é constituído por cerca de 250 empresas, concentradas fundamentalmente nas regiões de Marinha Grande (aprox. 60%) e Oliveira de Azeméis (aprox. 35%), emprega cerca de 7.500 trabalhadores (cerca de 30 trabalhadores por empresa). Este Sector apresenta um Valor Acrescentado superior a 80%, e exporta cerca de 90% das suas vendas, tendo demonstrado ao longo dos anos, uma elevada capacidade de adaptação às evoluções, quer dos mercados quer das tecnologias. Como entidades de suporte, este sector industrial apresenta como principais referências, a **CEFAMOL** - Associação Nacional da Indústria de Moldes (fundada em 1969), o **CENTIMFE** - Centro Tecnológico da Indústria de Moldes, Ferramentas Especiais e Plásticos (fundado em 1991) e o **CENFIM** - Centro de Formação Profissional da Indústria Metalúrgica e Metalomecânica (instituído em 1985).

Assim, a crescente articulação entre as empresas deste sector e aquelas Instituições muito tem contribuído para que a produção nacional de Moldes, seja hoje em dia, um dos sectores mais competitivos da Indústria portuguesa a nível internacional. O progresso e a vanguarda desta Indústria reconhecida internacionalmente, deve-se para além da sólida experiência e *Know-how* (quer em projecto quer em fabricação), ao cumprimento dos prazos de entrega, ao rigoroso controlo de qualidade, à elevada experiência, à competitividade e investimento em alta tecnologia, factores que asseguram a continuidade do fornecimento de Moldes portugueses aos mercados mais exigentes do mundo.

A primeira grande ferramenta utilizada no âmbito deste projecto, foi o Microsoft Visual Basic que se encontra disponível dentro do CATIA V5 na forma de um editor de VBA dedicado à personalização da ferramenta às necessidades do cliente. Em segundo lugar, será abordada outra forma de automatizar o desenho, sendo esta uma ferramenta que existe dentro do CATIA que é denominada por *Knowledgeware*, cuja função será descrita mais à frente.

O objectivo a atingir neste trabalho reside na definição de componentes comuns num molde, capazes de se adaptar ao contexto em que se inserem e a introdução de lógica de família e de standard, para a racionalização de componentes a fabricar. A concepção de uma aplicação com este fim requer um estudo prévio, para determinar as necessidades mais imediatas neste tipo de indústria, para depois modelar a aplicação conforme os requisitos.

A necessidade de cumprir prazos muito apertados, que é uma constante neste tipo de Indústria, justifica que se explorem formas de automatizar todo o processo de concepção dos Moldes, em especial na fase de modelação através da ferramenta de desenho em causa. Desta forma, justifica-se que seja possível definir componentes que resultam de tarefas que são idênticas em todos os Moldes que as empresas produzem, e desenvolver ferramentas que possam encurtar o tempo em excesso que resulta da sua repetição.

A CadTech Ibérica, S.A.

2

2. A CadTech Ibérica, S.A.

O crescimento da indústria de componentes para automóveis em Portugal motivou o crescimento de diversas empresas que, directa ou indirectamente, se relacionam com o sector. A CadTech é um desses exemplos. O aparecimento da CadTech está associado a dois factores principais; o primeiro prende-se com uma repentina e dominante implementação no mercado mundial do principal software com que esta empresa trabalha, o CATIA. Este software, num espaço de 5 anos passou a ser a ferramenta CAD/CAM/CAE de eleição para projecto e simulação digital, pela quase totalidade dos fabricantes ligados à indústria automóvel.

Paralelamente a este fenómeno surge o segundo factor, relacionado com o crescente envolvimento dos fornecedores da Indústria Automóvel no desenvolvimento dos componentes que fabricam, ou seja, o construtor automóvel com vista a encurtar os prazos de projecto e de fabrico, beneficiando simultaneamente do *know-how* específico dos seus fornecedores, começou a exigir a estes que participassem activamente no desenvolvimento das peças que iriam fabricar. Este facto levou a que as empresas portuguesas, fabricantes de componentes automóveis, necessitassem de um parceiro na área das novas tecnologias, que lhes proporcionasse acesso às mesmas ferramentas e métodos que os seus clientes utilizavam e exigiam.

A CadTech define-se como um integrador de soluções CAD/CAM/CAE/PDM e VR (Virtual Reality) com a missão de ajudar as empresas industriais a serem mais eficazes e competitivas, através da utilização de ferramentas de desenho, simulação, cálculo, gestão documental e realidade virtual. Estas soluções apoiam-se fundamentalmente em duas famílias de software, das quais a CadTech é o maior e mais antigo fornecedor em Portugal. Na primeira encontram-se a linha de produtos da IBM/Dassault : CATIA, SmarTeam, Delmia e Enovia. Na segunda incluem-se todos os produtos Autodesk para o projecto industrial: o AutoCAD, AutoCAD Mechanical, Mechanical Desktop e Inventor. Pessoalmente, tive a oportunidade de aprender e desenvolver ferramentas para o CATIA V5, em especial os módulos MD2 (Mechanical Design), CCV (Core & Cavity Design) e o MTD (Mold Tooling Design), sendo estes dois últimos vocacionados em especial para a Indústria de Moldes.

Complementarmente, esta empresa representa também os produtos da Hungsberg e ICD para comunicações EDI (Daxware e Cayor), o software da PathTrace (Edge Cam) para fresagem e torneamento, o software da FCC para chapa metálica (Autopol), o software da MsC para cálculo por elementos finitos (Nastran, Patran, Marc e Visual Nastran) e outros aplicativos de suporte como conversores (Transcat e Cadcam-E), visualizadores (3Dview e Wiseview) ou bibliotecas de componentes (TraceParts).

CADTECH

PREMIER PARTNER 3DS CATIA

Advanced PARTNER 3DS CATIA

CATIA Version 5 Release 10

3

3. CATIA Version 5 Release 10

3.1 Curso de formação em CATIA V5: MD2 – Mechanical Design

O primeiro passo para o início do desenvolvimento deste Projecto de Fim de Curso, foi ter tido a oportunidade de participar num curso de formação de iniciação em **CATIA V5**. Este curso teve lugar nos escritórios da CadTech em Lisboa e teve a duração de 5 dias. Este curso foi fundamental para o conhecimento da ferramenta com a qual iria trabalhar, pois sem um conhecimento profundo do CATIA certamente não seria possível desenvolver aplicações dedicadas para utilizadores experientes. Para isto era necessário conhecer as potencialidades deste tipo de software e em especial, o módulo dedicado aos Moldes que será referido mais à frente neste texto.

No final do curso, os assistentes estarão em condições de conhecer as funções e metodologia necessária para o desenho de modelos em CATIA V5. Este curso contém os seguintes módulos: *Sketcher*, *Part Design*, *Assembly Design*, *Wireframe* & *Surface*, *Generative Shape Design*, *Interactive Drafting*, *Generative Drafting*, *Real Time Rendering* e conceitos gerais de interoperabilidade CATIA V4 → V5.

CONTEÚDO DO CURSO:

Sketcher & Part Design:

1. Conceitos de modelação em sólidos
2. Uso da ferramenta Sketcher para geração de perfis
3. Sólidos baseados em perfis do Sketcher (pad, pocket, rib, lofts...)
4. Ferramentas de "dress-up" de sólidos (fillets, drafts...)
5. Repetição de features e sólidos (patterns, mirrows...)
6. Uso e aplicação de operações boleanas
7. Modelado híbrido: Uso de superfícies para a definição de sólidos
8. Metodologia de modificação de sólidos já existentes
9. Análises de propriedades da geometria: Medidas, peso e momentos de inércia.
10. Parametrização de sólidos e criação de sólidos mediante tabelas de Excel
11. Criação de Anotações 3D

Assembly Design:

1. Conceitos de uso de Assembly
2. Uso de estrutura Product como modelo habitual do desenho
3. Uso e aplicação de restrições entre peças
4. Operações de conjuntos: "Assembly features"
5. Gestão e manipulação de Parts e Products
6. Modificação de restrições de conjuntos
7. Análises de restrições, dependências e graus de liberdade
8. Geração de explosões
9. Criação de conjuntos "flexíveis"
10. Assemblies simétricos
11. Ferramentas DMU de inspecção: Cálculo de interferências, seccionando dinâmico e análise de distâncias

Wireframe & Surface:

1. Conceitos de trabalho com superfícies e geometria wireframe
2. Estratégia de trabalho no modelo híbrido
3. Geração de geometria wireframe: pontos, linhas, planos, cónicas, splines, hélices, polilinhas...
4. Geração de superfícies: extrusão, revolução...
5. Criação de superfícies avançadas: lofts, blends, sweeps...
6. Geração de curvas a partir de outros elementos: projecções, intersecções, curvas limite, connects...
7. Manipulação de superfícies: trim , split, join, healing...
8. Ferramentas de transformação de superfícies: simetrias, translações, escalas...
9. Geração de sólidos a partir de superfícies e modificação de superfícies.

Generative Drafting:

1. Geração de vistas frontais, projecções, vistas isométricas e vistas auxiliares.
2. Geração de secções e cortes assim como secções e cortes rebatidos
3. Criação de detalhes, roturas e roturas parciais
4. Geração de vistas automáticas.
5. Cotação manual, automática e passo a passo dos planos
6. Criação de textos, anotações e símbolos.
7. Modificações geométricas e gráficas dos elementos gerados
8. Gestão de standard de desenho
9. Criação e uso de detalhes e catálogos 2D
10. Geração de geometria usando ferramentas 2D
11. Uso e criação de formatos e plotagem de planos

Interactive Drafting:

1. Geração de geometria: pontos, linhas, círculos, arcos...
2. Cotagem da geometria gerada
3. Criação de textos, anotações, tabelas e símbolos.
4. Modificações geométricas e gráficas dos elementos gerados
5. Gestão e criação de formatos e plotagem de planos

V4 Interoperability:

1. Conceitos gerais de interoperabilidade V4 / V5
2. Leitura de ficheiros V4 sem conversão
3. Conversão de ficheiros V4 e V5, passos a seguir e ferramentas de verificação.
4. Conversão de ficheiros de V4 e V5 e passos a seguir.
5. Leitura de ficheiros V4 em modo "Batch".
6. Leitura de ficheiros V4 exportados.

Real Time Rendering:

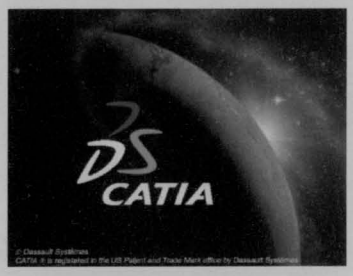
1. Aplicação de materiais da livraria standard.
2. Personalização de livrarias de materiais.

Duração:

60 horas.

3.2 Metodologia de Modelação 3D

A metodologia utilizada no *Part Design* do CATIA V5, é de uma forma geral similar à dos softwares de modelação 3D existentes no mercado. De facto, o CATIA marca a diferença na facilidade de execução de operações, na simplicidade de funcionamento das diferentes funcionalidades, sendo estas qualidades aliadas a um ambiente de trabalho bastante agradável e muito equilibrado, equipado com barras de ícones que descrevem quase ao pormenor a operação que desenvolvem. Estas qualidades traduzem-se num encurtar do tempo necessário para modelar uma peça em comparação com qualquer outro software.



O ambiente de trabalho do CATIA V5 é composto por três elementos fundamentais na modelação 3D, para além das barras de ferramentas, entre os quais:

- **Árvore de Especificações (1):**

  Este elemento é fundamental, visto que é a partir daqui que toda a organização da sequência de operações é estruturada de maneira a facilitar o acesso a operações anteriores, possibilitando a sua edição. Em programação, e como vamos poder ver mais à frente, esta estrutura organizacional é fundamental, visto que é a partir daqui que toda a sequência de procedimentos é desenrolada (ver figura 3).

- **Planos de Sketch (2):**

  Estes elementos são fundamentais, visto que é através destes elementos que designamos o referencial de base do sistema e o plano em que se insere. O centro deste sistema de eixos planar será o (0,0,0) de todas as nossas peças (ver figura 3).

- **Compasso (3):**

  Este elemento situa-se sempre no canto superior direito do nosso ambiente de trabalho e é importante para a execução de operações de *Assembly* de várias peças quando se forma um Produto (ver figura 3).

Nesta fase é importante realçar que a estrutura da árvore é diferente para uma peça (denominada por *Part*), e para uma estrutura de um conjunto de peças (ou Produtos) que formam um Produto (Product).

Fig. 1 – Estrutura base de uma *Part*

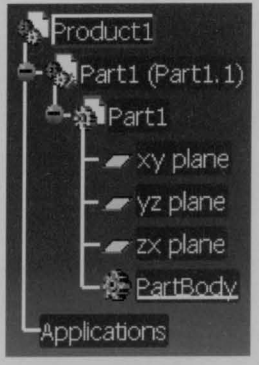


Fig. 2 – Estrutura base de um *Product*

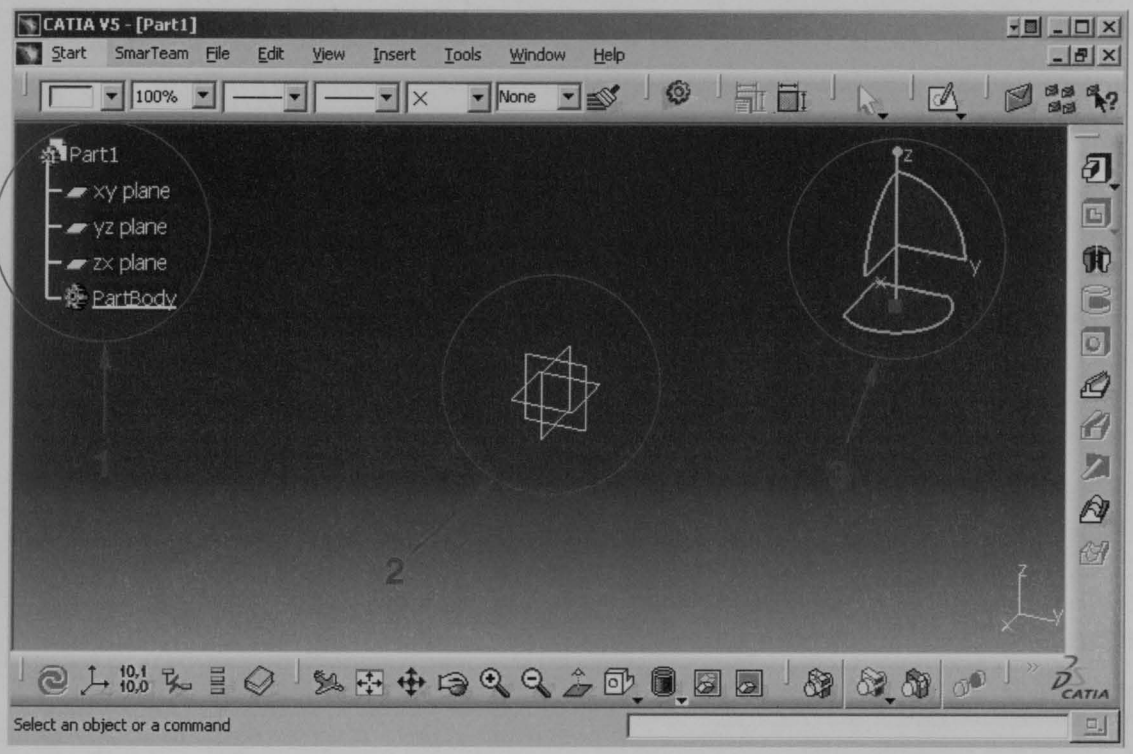


Fig. 3 – Ambiente de trabalho CATIA V5

Depois dos elementos mais básicos do ambiente de trabalho, chegou o momento de passarmos a descrever as diferentes ***Workbenches*** (bancadas de trabalho). O CATIA é composto por várias *Workbenches* que nos são apresentadas no menu principal → *Start* (figura 4 e 5); no âmbito deste trabalho foram utilizadas as seguintes *Workbenches*:

Mechanical Design:

- Part Design
- Assembly Design
- Mold Tooling Design
- Core & Cavity Design
- Drafting

Shape:

- Generative Shape Design

Knowledgeware:

- Knowledge Advisor
- Knowledge Expert
- Product Engineering Optimizer
- Product Knowledge Template

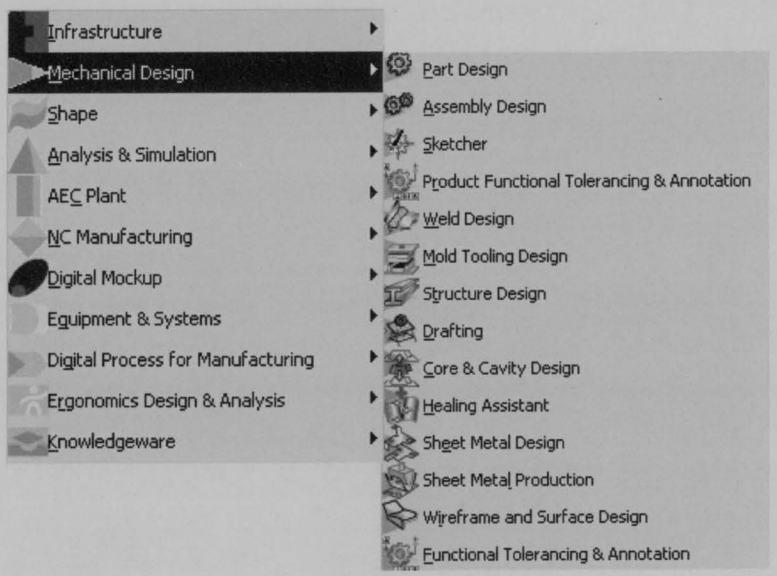


Fig. 4 – Workbench Mechanical Design

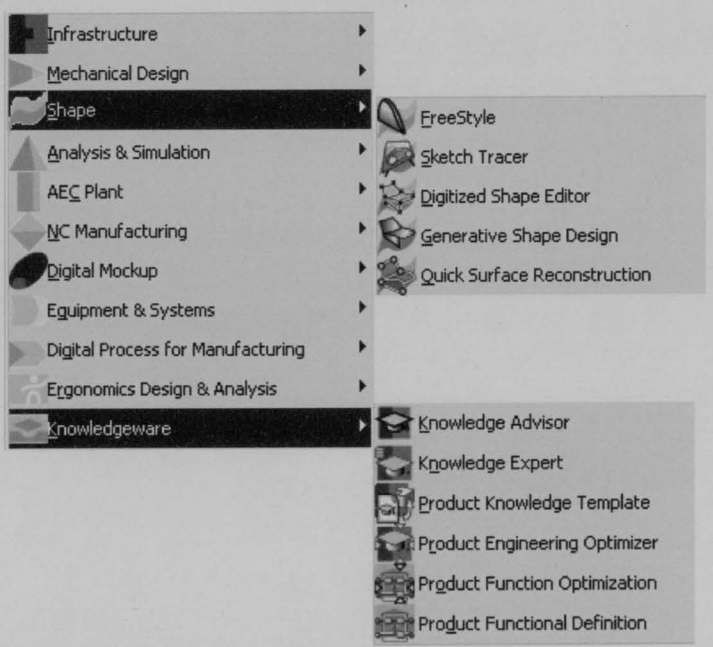


Fig. 5 – Workbench Shape e Knowledgeware

A metodologia utilizada é composta pela criação de um *Sketch* (esboço) preliminar elaborado em 2D, no qual estão atribuídas aos elementos geométricos, todas as restrições necessárias para a obtenção do objecto pretendido. Estas restrições podem ir desde as restrições geométricas como paralelismos, perpendicularidades, coincidências, etc., até às restrições dimensionais como são as cotas. Depois desta fase, passamos à modelação em 3D na qual podemos efectuar todo o tipo de operações como *Pad* (Extrusão de um perfil com uma determinada Altura), Furos, Chanfros, Boleados, etc.

Para a correcta utilização de todas as potencialidades do CATIA V5, é necessário entender os princípios e fundamentos pelos quais esta ferramenta se rege. Vamos começar pela lógica da *Part*.

Uma *Part* é composta por um *Part Body*, que neste caso é o nosso primeiro corpo. Este corpo encerrará em si próprio todas as operações que serão efectuadas com o Sólido, ou seja, só conterá dentro de si toda a informação referente a o Sólido. Por outro lado, existe também o *Open Body*. Este corpo conterá toda a informação referente a todas as outras operações que não são directamente executadas no sólido, por exemplo, podem existir referências de planos, pontos, arestas de outras *Parts* que necessitemos de aproveitar para a execução da nossa *Part*. Este corpo vai conter também todas as superfícies e todas as operações executadas com o GSD (Generative Shape Design) que será abordado mais à frente. Por exemplo, quando necessitamos que um corpo sólido tenha a sua origem a partir de uma superfície, este corpo será chamado *Hybrid Body* (Corpo Híbrido) e toda a sua informação será guardada no *Open Body*. É de realçar que este *Open Body* só aparecerá se de facto tivermos a necessidade de executar este tipo de operações, se só estivermos a trabalhar com o Sólido, apenas vamos ter o *Part Body*.

Portanto, para ter uma *Part* vamos ter vários *Bodies* (Part Body, Open Body, etc.). Isto não impede que na mesma *Part* possamos ter outros *Part Bodies*, o que interessa realçar e que tudo o que estiver dentro da *Part* (figura 6) formará uma só peça com vários corpos.

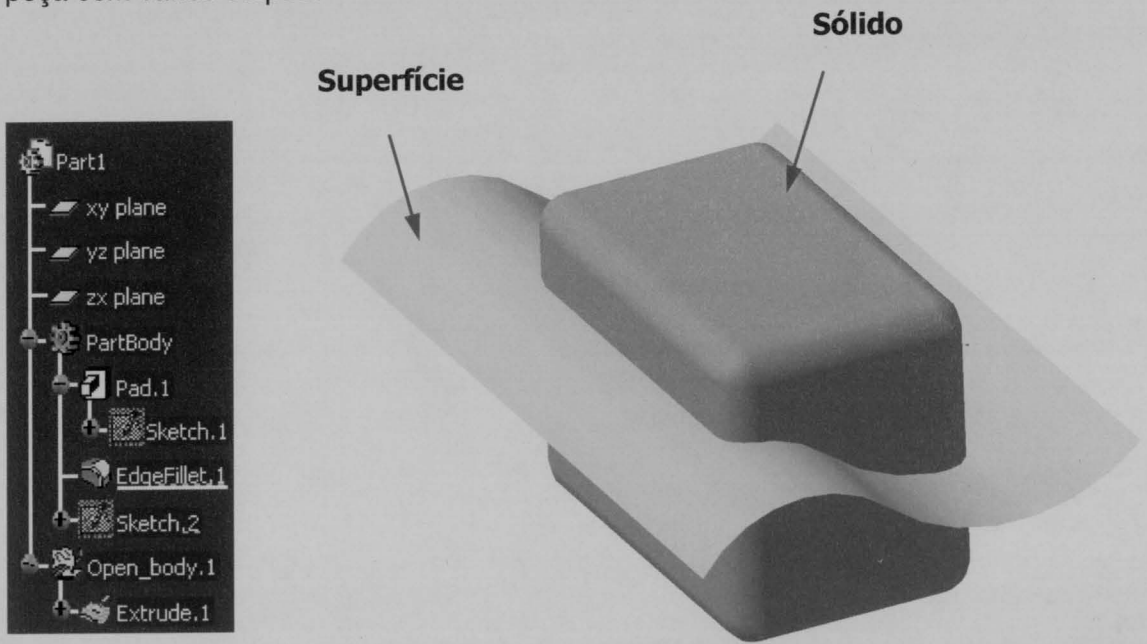


Fig. 6 – Estrutura de uma *Part* contendo vários *Bodies*

Pode existir a necessidade de termos na estrutura da nossa peça, a presença de um corpo "negativo". Este corpo negativo que terá o nome de ***DrillHole***, e não é mais do que um corpo idêntico ao que temos, mas com a função de abrir espaço dentro de um componente para este corpo (vamos poder ver esta situação descrita na sua plenitude mais à frente, quando realizarmos os Postiços). A semelhança dos *Part Body*, também podemos ter vários *Open Body*.

Mais à frente, vamos poder perceber a importância destes conceitos na elaboração das aplicações, em especial a que se refere à criação de Postiços.

É necessário referir a importância de se trabalhar com o "Histórico". Se olharmos para a estrutura da árvore, podemos verificar que está implícita a ideia de "Histórico", que contempla uma estrutura ramificada na qual podemos encontrar a ligação entre as várias operações efectuadas e temos o privilégio de poder modificar, ou mesmo, apagar qualquer operação. Se pretendermos modificar uma operação, vamos poder verificar que as operações que estão a jusante, vão ser actualizadas. Isto deve-se ao facto de as operações estarem ligadas através de um ***Link***, daqui resultando o facto de a modelação, em 3D, ser regenerativa.

Esta ideia é importante, visto que, pode existir a necessidade de quebrar esse *Link* entre as operações. Isto pode ser feito sem qualquer dificuldade, mas a regenerativadade do 3D e as suas vantagens serão perdidas. Por exemplo, vamos imaginar que temos um sólido que é cortado com uma superfície (Split); se pretendermos modificar a superfície, vamos poder verificar que à parte do sólido que foi "cortada" com essa entidade, vai ser actualizada. Esta função resulta da utilização do *Link*. Se optarmos por não trabalhar com histórico, não vamos poder actualizar o sólido (figura 7). Nos capítulos 8 e 9, vamos poder perceber a importância destas definições.

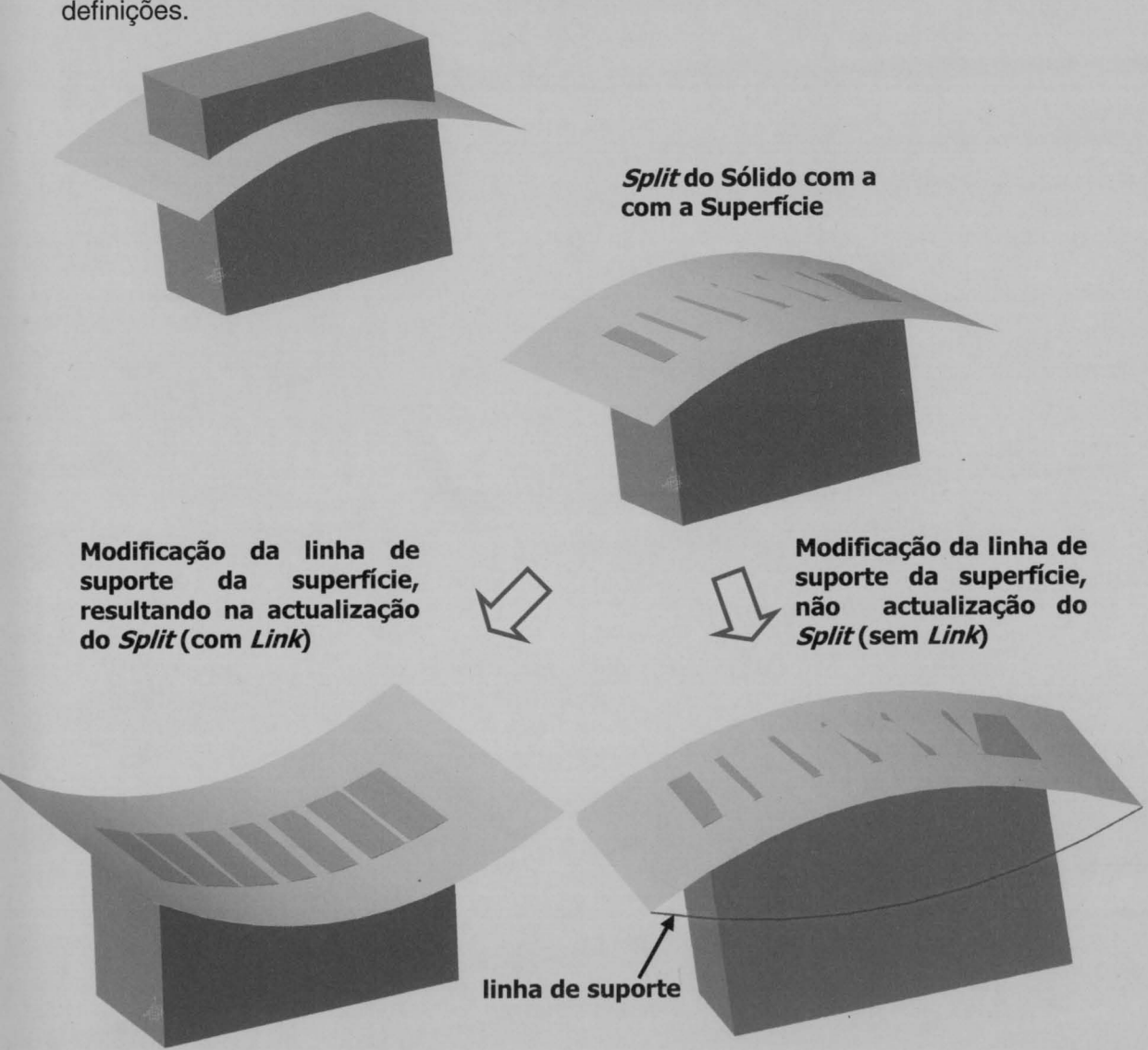


Fig. 7 – Definição da mesma operação com e sem *Link*

3.3 Generative Shape Design

Outra *Workbench* muito importante para a concepção de peças a moldar e seus respectivos Moldes, é o **GSD** (Generative Shape Design).

Como vamos poder verificar, o GSD dá-nos a possibilidade de conceber muito facilmente corpos complexos com a ajuda de superfícies, que podem surgir de uma estrutura *Wireframe*, ou podem resultar de um conjunto de superfícies. Na concepção de um Molde o conhecimento desta *Workbench* é fundamental, visto que a separação Macho / Cavidade será executada com recurso a uma série de superfícies que são geradas a partir do modelo da peça a ser moldada.

Entre as ferramentas que temos ao nosso dispor, podemos encontrar a geração de geometria *Wireframe*, como por exemplo pontos, linhas, planos, cónicas, *splines*, hélices, polilinhas, etc., ainda a geração de superfícies simples e complexas, como *lofts*, *blends*, *sweeps*, etc. A manipulação das superfícies também é um ponto forte do GSD, contendo ferramentas como o *Trim*, *Split*, *Join*, *Healing*, etc.

Uma das grandes vantagens do GSD é a sua capacidade para gerar corpos sólidos híbridos, obtidos a partir de superfícies. Esta capacidade advém da possibilidade de trabalhar com histórico (esta noção foi descrita no ponto anterior), sendo que a qualquer momento podemos fazer alterações na superfície que deu origem ao sólido, visto que as alterações são actualizadas automaticamente.

Dentro do CCV (Core & Cavity Design) existem as funcionaliades que facilitam a modelação das superfícies de partição a partir do modelo da peça a moldar.

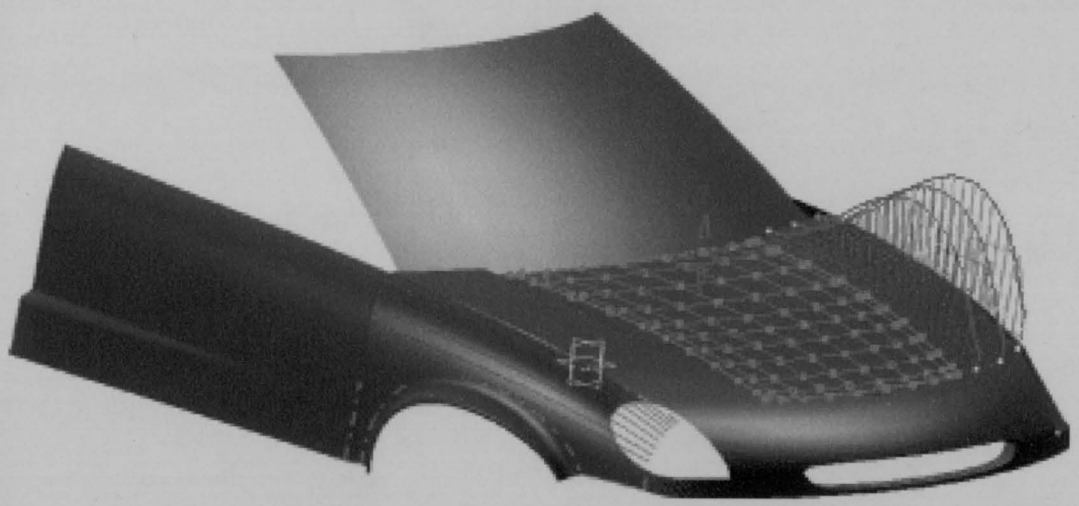


Fig. 8 – Superfície gerada através do *Generative Shape Design*

3.4 Interactive Drafting

O *Interactive Drafting* é uma ferramenta que executa o desenho 2D de uma peça modelada em 3D. Esta ferramenta consegue melhorar a produtividade visto que, à semelhança do GSD, consegue tornar o desenho técnico de uma peça completamente regenerativo. Esta vantagem advém do desenho 2D ser gerado directamente a partir do 3D, resultando numa ligação que nos permite, a qualquer momento, fazer alterações no modelo 3D e obter a actualização facilmente do desenho 2D.

Esta ferramenta também contempla as Operações / Instruções necessárias à correcta realização de um desenho técnico 2D.

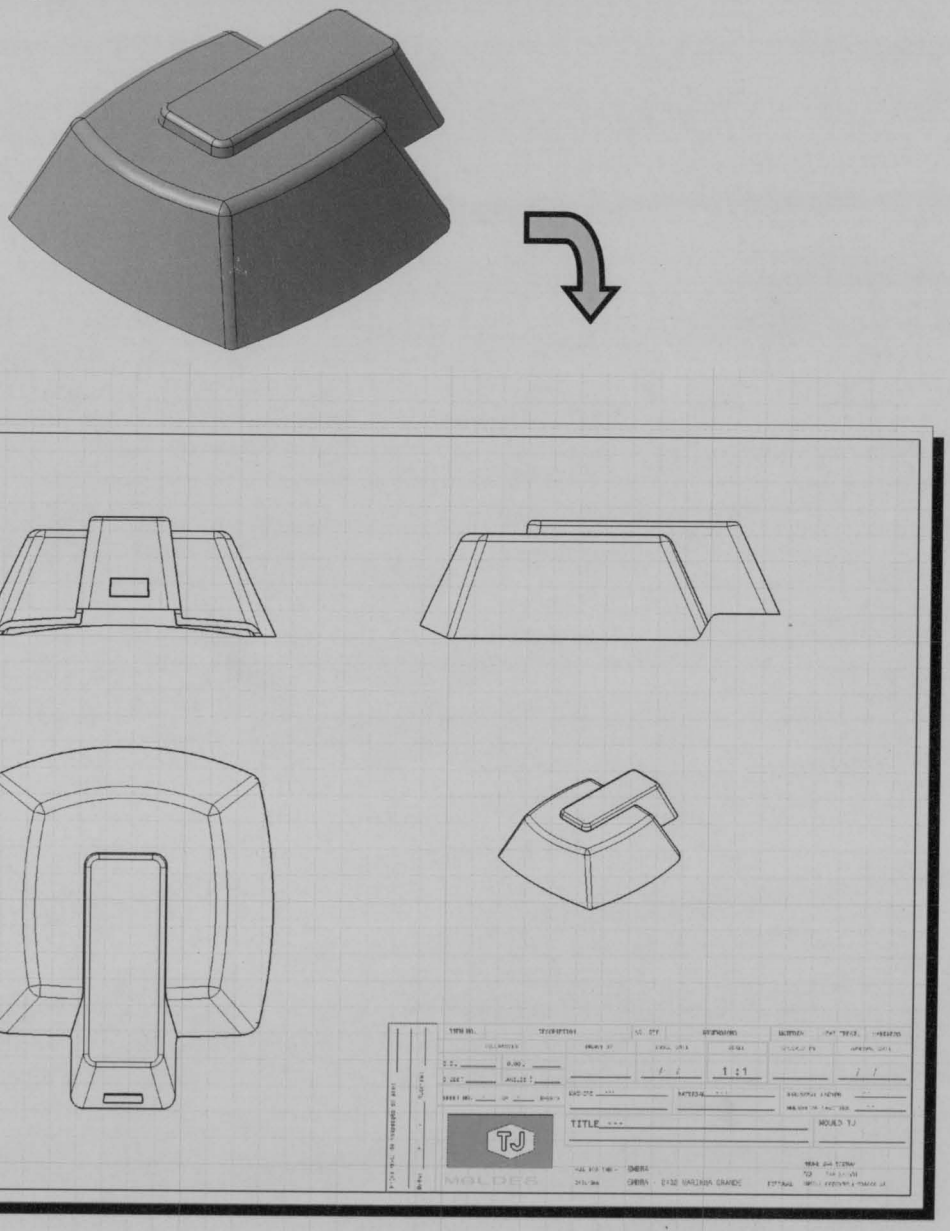


Fig. 9 – Desenho 2D gerado a partir de um modelo 3D

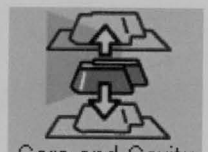
Core & Cavity Design

4

4. Core & Cavity Design

4.1 Metodologia de separação Macho / Cavidade

Esta *Workbench* consiste numa série de operações que definem a preparação da peça a moldar, para dar origem à separação das placas da estrutura do Molde que será executada na *Workbench Mold Tooling Design* (**MTD**). A metodologia da separação do Macho / Cavidade consiste na definição do sentido de extracção, direcção dos movimentos (se necessário), sendo que o **CCV** (Core & Cavity Design) está equipado com barras de ferramentas que existem no **GSD** (Generative Shape Design) para permitir obter as superfícies de partição. Estas superfícies de partição são definidas através de duas superfícies, que são denominadas por "peles", visto que uma representa a superfície da Cavidade e a outra representa a superfície do Macho.



A demonstração desta metodologia será feita através de um exemplo prático, representado pela separação Macho / Cavidade da peça seguinte:

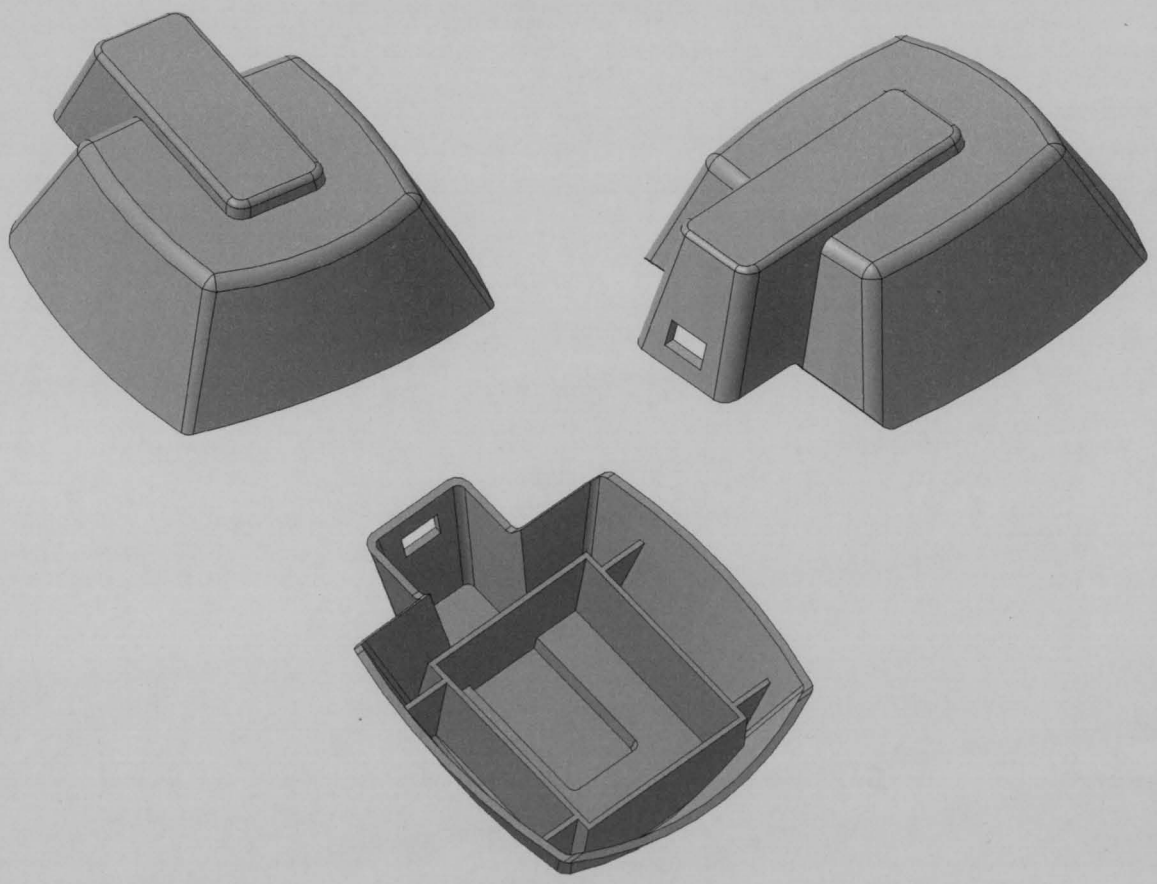


Fig. 10 – Peça a ser moldada

O primeiro passo a realizar, é importar a peça para dentro do Produto, que vai conter a peça a moldar (Molded Part) e a estrutura do Molde. Para dar origem a esta estrutura abrimos primeiro a *Workbench* **CCV** (Core & Cavity Design), onde vamos clicar no ícone ***Import Model***  , que provocará o aparecimento do seguinte menu:

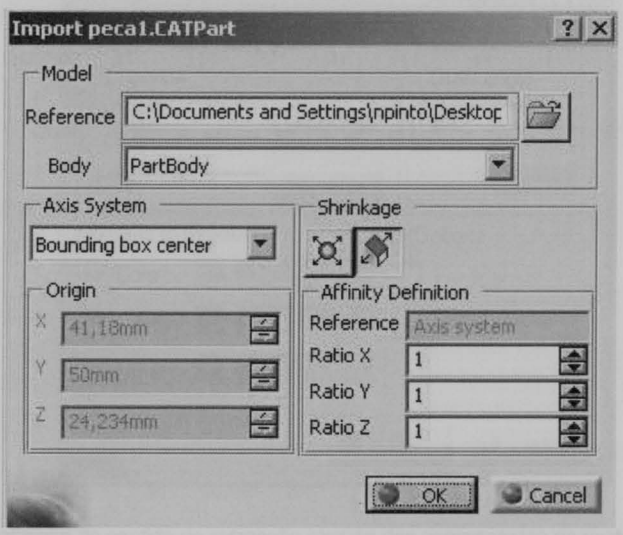


Fig. 11 – Menu para importação da Peça

Para trabalhar correctamente no *Mold Tooling Design*, é fundamental que seja seguida a estrutura original do MTD que nos é fornecida se efectuarmos as operações descritas correctamente. Isto é muito importante para a criação das aplicações que estão descritas mais à frente, visto que a personalização da ferramenta depende da forma de trabalhar do utilizador. Então, se o utilizador for organizado terá mais possibilidades de não ter problemas em operações futuras e a aplicação desenvolvida funcionará correctamente. Depois da validação do menu anterior, aparece a estrutura da árvore seguinte:

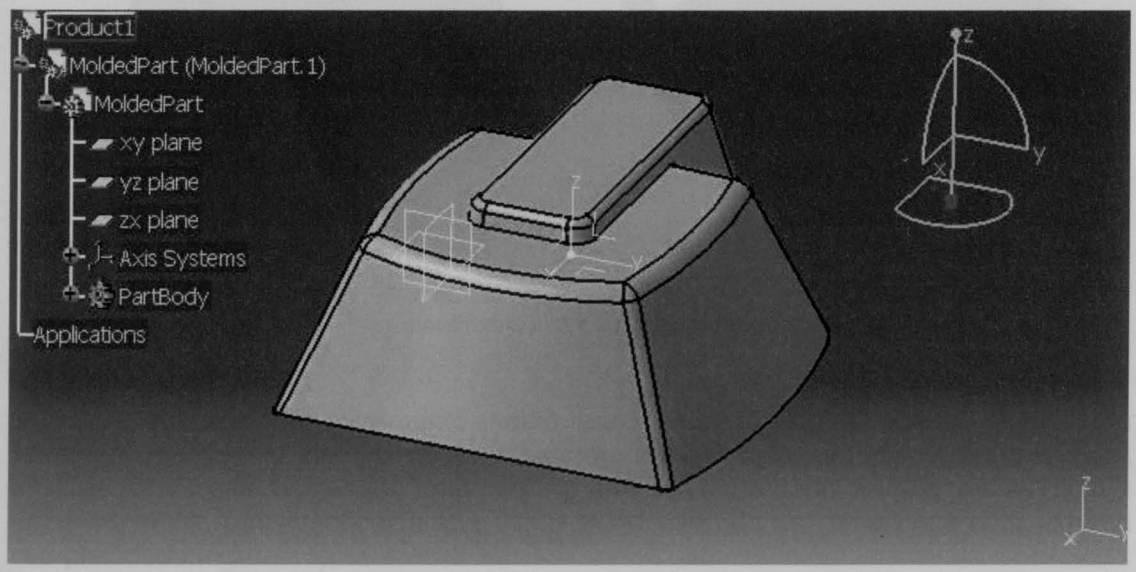


Fig. 12 – Produto com a Peça Importada

O passo seguinte será definir a direcção de extracção, sendo que aqui vamos necessitar do "Compasso". Para isto temos que clicar em ***Main Pulling Direction*** onde vai aparecer o seguinte menu:

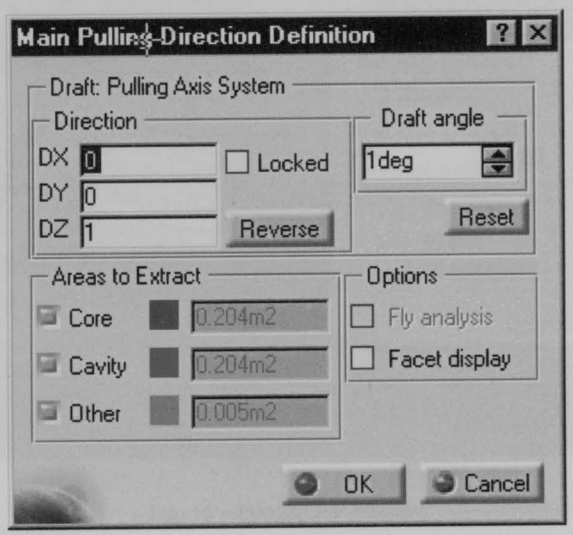


Fig. 13 – Definição do sentido de extracção através das componentes do vector DX, DY e DZ

Se clicarmos na peça podemos reparar que mudou de cor, representando a cor verde a zona do lado fixo, e a cor vermelha a zona do lado móvel. No MTD está predefinido que do lado móvel vai ficar o macho e do lado fixo, a cavidade; No entanto, esta situação poderá ser alterada em função da geometria da peça.

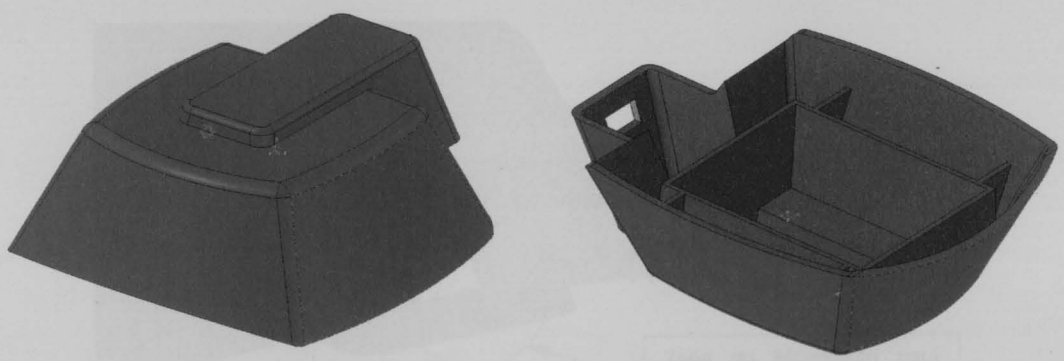


Fig. 14 – Peça com a definição dos lados do Macho e da Cavidade

A direcção do movimento neste caso é segundo **zz**. Existe ainda uma ferramenta adicional dentro deste comando denominada de ***Fly Analisys***. Esta ferramenta permite-nos fazer uma análise pormenorizada do ângulo de saída em todas as faces, para verificar a sua validade no caso de ser uma peça com alguma complexidade (figura 15).

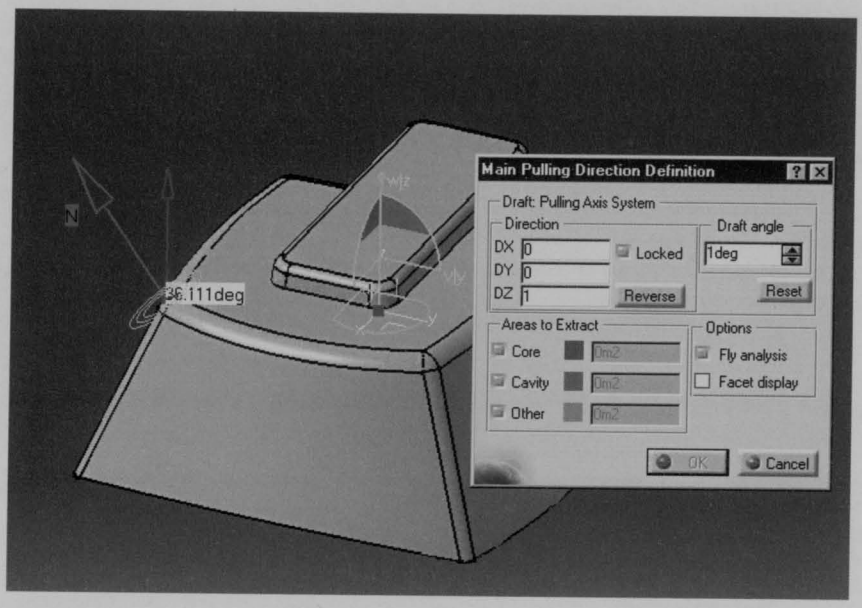


Fig. 15 – Função *Fly Analisys*

No caso desta peça, vai haver a necessidade de definir um movimento. Como representa a figura 16, esta será a zona que teremos que trabalhar. Podemos notar ainda que a cor adoptada para as faces a serem extraídas, não é nem verde nem vermelho, mas sim azul claro. O MTD automaticamente nos sugere quais as faces a serem extraídas para executar o movimento, no entanto, como no passo anterior, estas faces podem ser definidas de outra forma, dependendo da geometria da peça. Para isso podemos executar o comando ***Transfer an Element***  , que nos possibilita adicionar e remover alguns destes elementos da selecção.

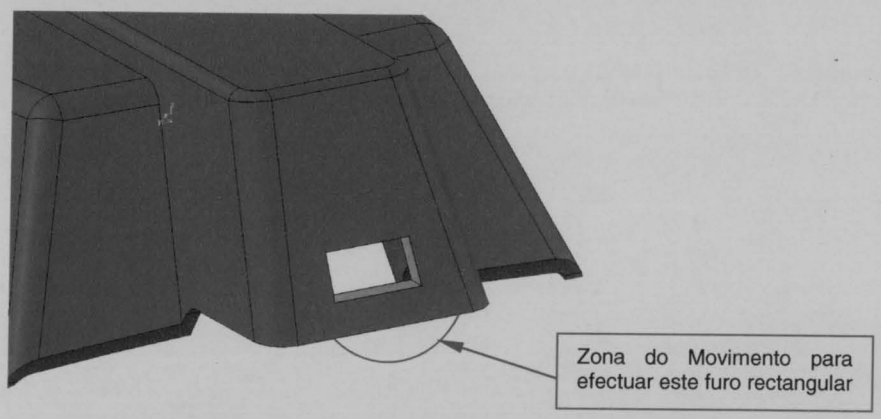


Fig. 16 – Superfícies (a azul claro) da zona onde será necessário definir um movimento

Visto não ser fundamental para este projecto a inclusão de movimentos, não irá ser explicada esta sequência de operações que serão importantes para a posterior definição do Molde. A separação das placas do molde faz parte da *Workbench* MTD que será explicada no próximo capítulo.

Mold Tooling Design

5

5. Mold Tooling Design

5.1 Preparação de uma Peça a ser Moldada

Antes de entrar na concepção do Molde, vamos numa primeira fase preparar uma peça para ser moldada. Esta operação consiste em criar as superfícies de separação a partir da separação do Macho / Cavidade feita no capítulo anterior. Estas superfícies vão ser obtidas através da *Workbench* GSD (Generative Shape Design).

O primeiro passo é carregar a peça que previamente já foi tratada com o CCV, e abrir o GSD recorrendo ao menu *Start*. A peça é representada na figura 17.

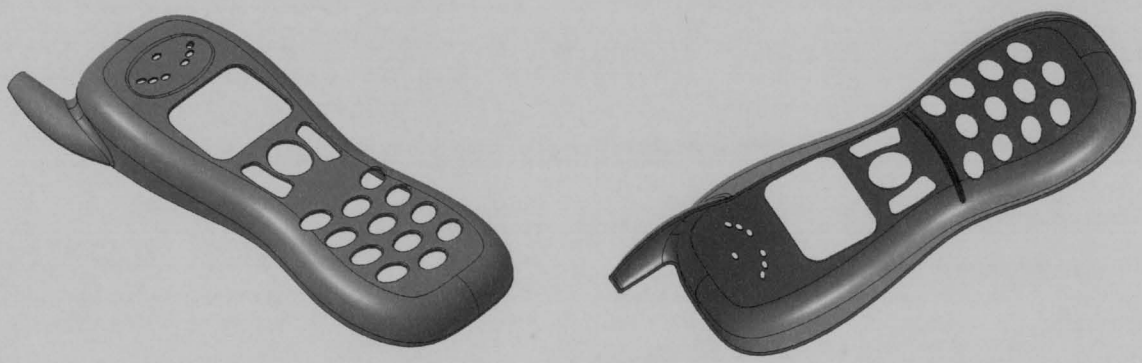


Fig. 17 – Peça a moldar

Quando abrimos o GSD, é criado um novo corpo com o nome "***Openbody.1***". Este novo corpo representa todas as superfícies que serão criadas a partir do *PartBody* (sendo este *PartBody* o objecto que contêm o sólido da peça). Aqui podemos editar as propriedades deste novo corpo e renomeá-lo como *PartingBody*, pois será o nosso corpo de partição.

Vamos começar por seleccionar todas as linhas exteriores da peça, como representado na figura 18, para definir a nossa linha de partição. Depois de seleccionadas, vamos utilizar o comando ***Join***  .

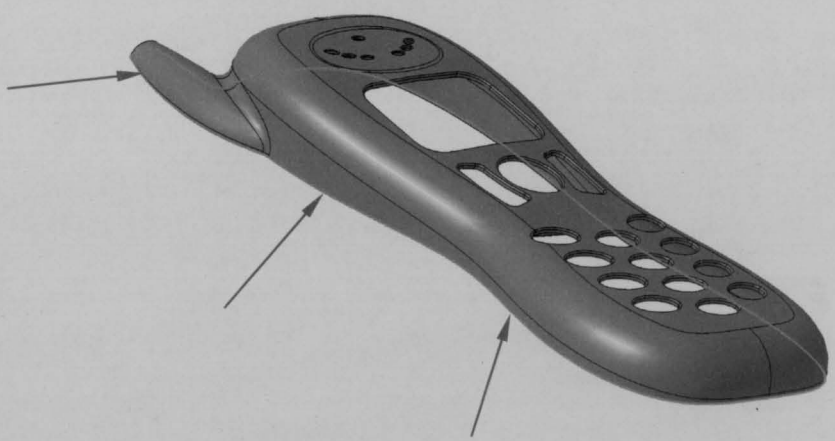


Fig. 18 – Conjunto de linhas que formam no seu conjunto a linha de partição

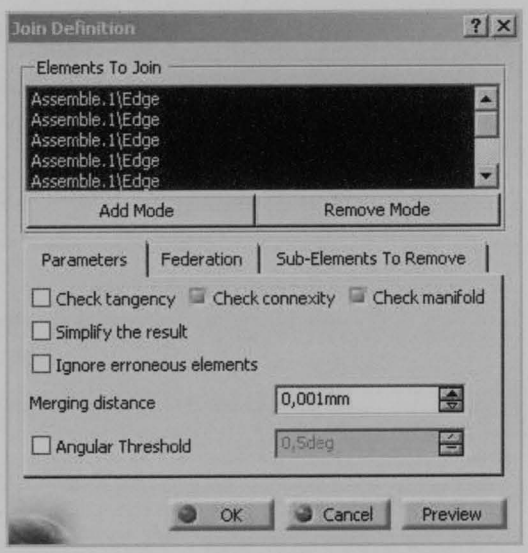


Fig. 19 – Agrupamento do conjunto de linhas para formar a linha de partição através do *Join*.

No menu da figura 19 é mostrado na caixa ***Elements to Join***, todos os elementos que seleccionamos. Fazendo ***OK***, construímos uma linha que resulta de todos os elementos que formam a linha de partição. Depois desta operação, passamos a tapar os furos para os botões e o ecrã. Para isso, executamos exactamente as mesmas operações descritas atrás, mas seleccionamos as linhas que formam a base dos furos como descrito na figura 20.

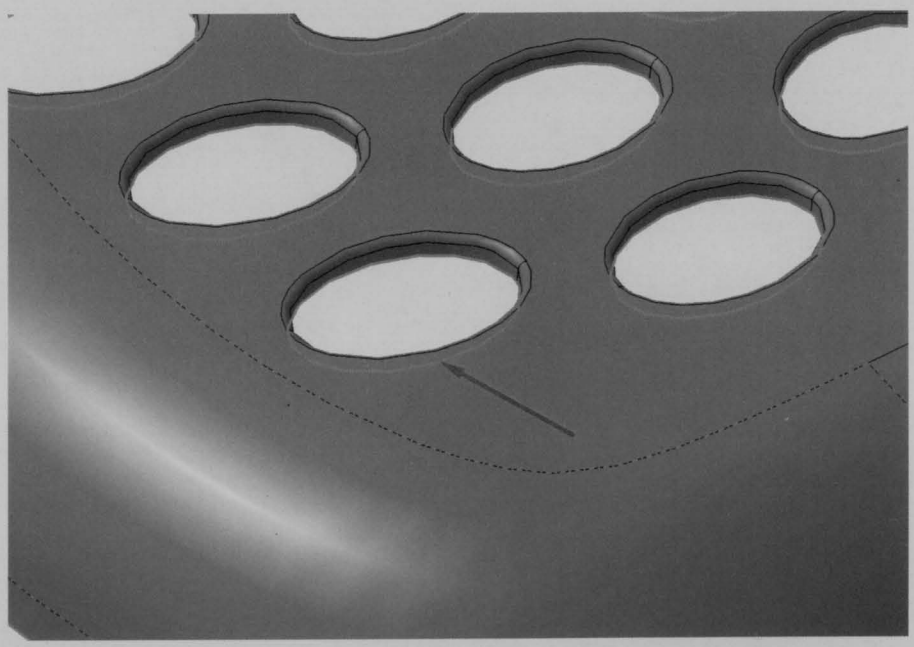


Fig. 20 – Linhas que formam a base dos furos a ser tapados

Agora vamos dar início à criação das superfícies de partição, sendo que em primeiro lugar, vamos criar as superfícies que vão tapar os furos. Para isso usamos o comando *Fill* e seleccionamos as linhas criadas anteriormente com o *Join*. Executamos esta operação para todos os furos (figura 21).

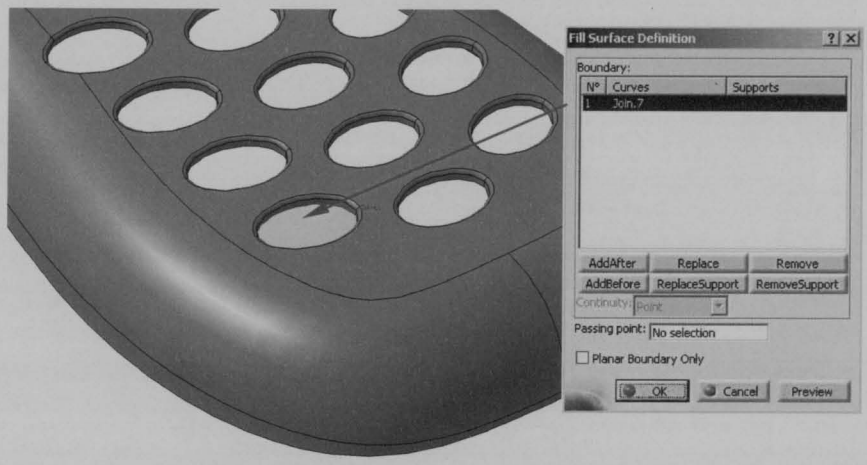


Fig. 21 – Utilização do *Fill* para criar as superfícies necessárias para tapar os furos

Para extrair as superfícies exteriores, vamos utilizar o comando ***Sweep*** e com as linhas extraídas em operações anteriores, vamos criar todas as superfícies de partição à volta da peça. É de realçar que se a peça for muito simples, esta operação poderá ser feita de uma vez utilizando toda a linha de partição; mas se for mais complexa, será boa prática criar uma superfície com cada linha de partição e depois fazer um *Join* seleccionando todas essas superfícies. As opções a seleccionar no menu *Sweep* estão apresentadas na figura 22.

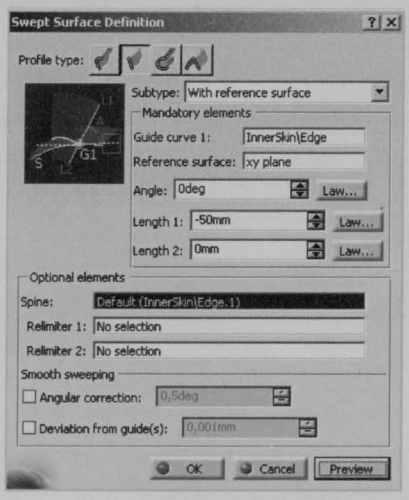


Fig. 22 – Utilização do *Sweep* para criar a superfícies de partição

Eventualmente, haverá a necessidade de fazer um *Join* com este novo conjunto de superfícies criado, visto que esta superfície terá que ser uma só e não poderá conter no seu domínio, qualquer tipo de abertura. Depois de efectuadas estas operações, vamos extrair as superfícies interiores e exteriores da peça. Para este efeito existe o comando ***Extract***  que extrai automaticamente estas duas superfícies (figura 23).

Depois da extracção destas duas superfícies, passamos a fazer outro *Join* para unir a superfície de partição com a superfície exterior da peça, e outro *Join* para unir a mesma superfície de partição com a superfície interior da peça.

Finalizando estas operações, estamos em condições de começar a construção da estrutura do Molde.

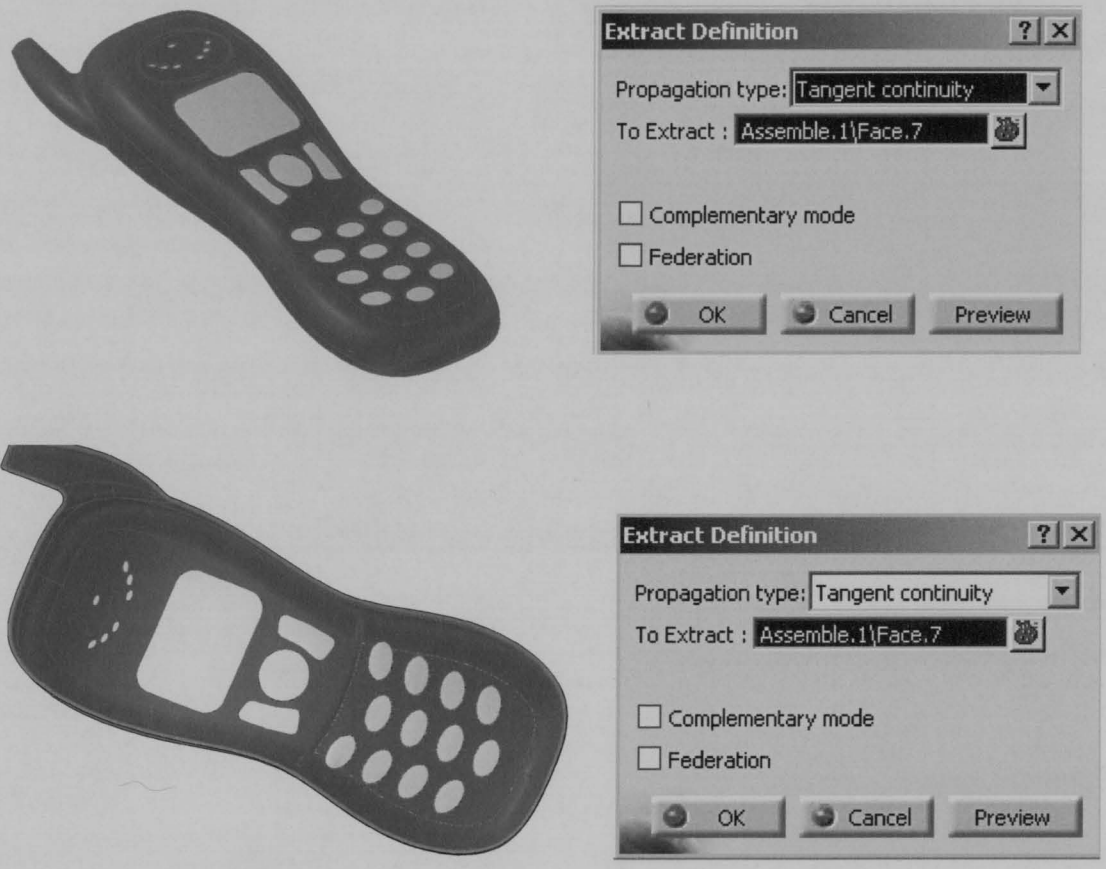


Fig. 23 – Extracção das duas superfícies que formam a "pele" da peça a ser moldada

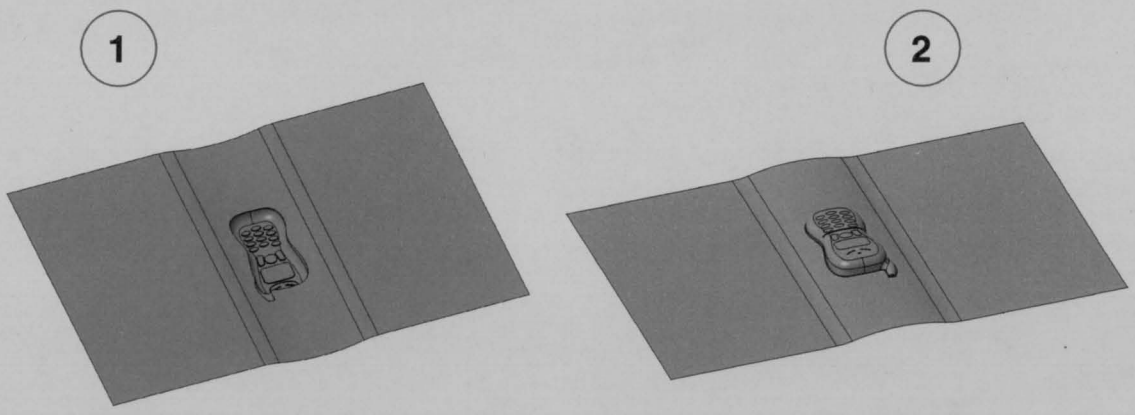


Fig. 24 – Superfícies de partição: (1) - Cavidade, (2) - Macho

5.2 Concepção da estrutura do Molde

Nesta nova etapa, vamos conceber a estrutura do Molde. Para efeito deste projecto, não vamos entrar na introdução de componentes no Molde como Guias, Extractores, Injector, Anel de Centragem, Linhas de Agua, Parafusos, etc. Na personalização da ferramenta que será descrita mais à frente, será apenas fundamental ter a estrutura da árvore correspondente ao MTD (Mold Tooling Design) correcta, visto que as aplicações desenvolvidas dependem apenas da estrutura do Molde como poderá ser visto e explicado mais à frente.

Vamos começar por abrir o nosso Produto, que como vimos atrás, representa o nosso Molde. Clicando duas vezes no nome do produto, entramos automaticamente na *Workbench* MTD. Aqui vamos clicar no comando ***Create a New Mold***  , onde nos vai aparecer o seguinte menu:

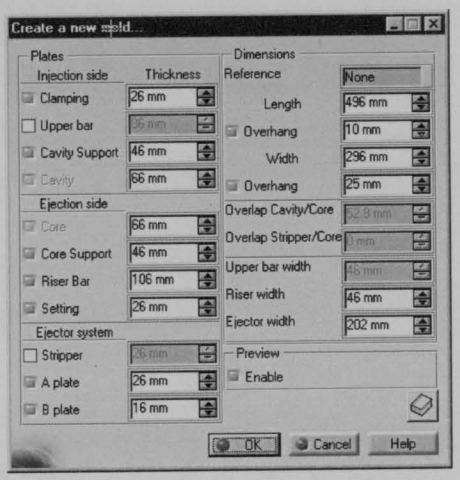


Fig. 25 – Menu utilizado para definir a estrutura

Neste menu podemos modificar as dimensões da estrutura e ainda, podemos seleccionar uma estrutura normalizada fornecida por um fabricante. Para isso basta clicarmos no ícone  , onde nos vai aparecer o menu da figura 26. Neste menu podemos seleccionar o fabricante, bem como as dimensões da estrutura.

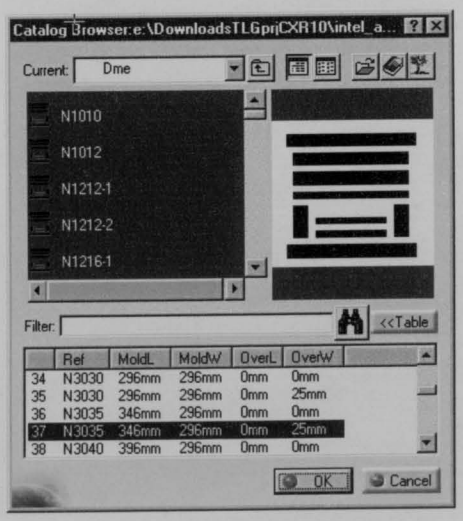


Fig. 26 – Catálogo de estruturas normalizadas de cada fabricante

Seleccionando uma estrutura à nossa escolha, vamos poder ver o aspecto que irá ter a estrutura do Molde (figura 27). Se fizermos ***OK***, finalizamos a operação aceitando a estrutura seleccionada.

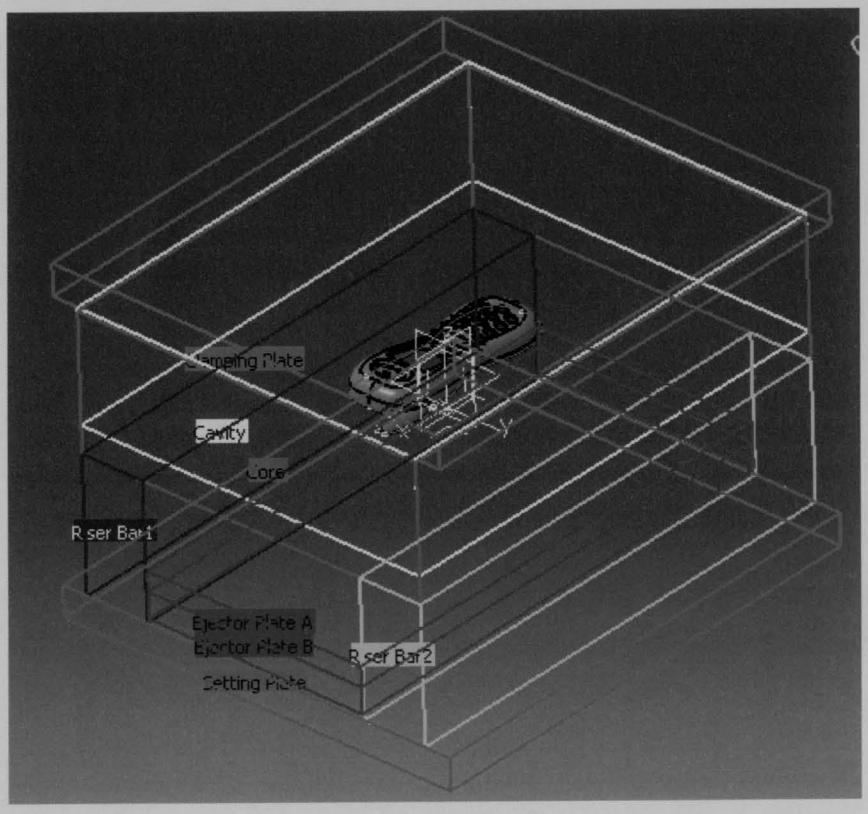
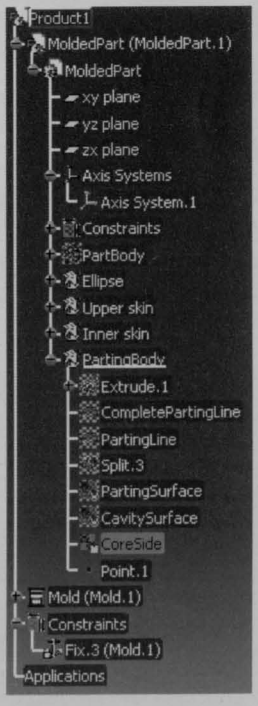


Fig. 27 – Aspecto preliminar da estrutura do Molde



Para centrar a peça no molde, podemos utilizar o comando ***Snap*** , onde vamos fazer com que o sistema de eixos da peça, seja o mesmo do Molde. É altura de separar as duas placas com as superfícies de partição que definimos anteriormente. Esta operação será executada seleccionando na árvore a placa com o nome *CorePlate*, que representa a placa do lado Móvel e com o botão direito do rato acedemos a um menu onde seleccionamos ***Split Component*** (figura 29). Com este comando vamos dar início à separação, fazendo um *Split* da placa com a superfície, que no caso da *CorePlate*, será a superfície com o nome *CoreSide* que podemos encontrar dentro da *MoldedPart*, como representa a figura 28. A mesma operação será efectuada na placa do lado Fixo, designada por *CavityPlate* e a superfície designada por *CavitySurface*.

Fig. 28 – Estrutura base do Produto que é o nosso molde.

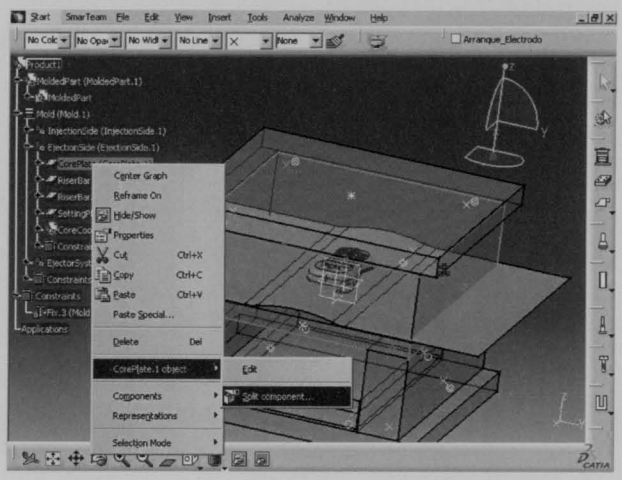


Fig. 29 – Menu onde podemos encontrar o *Split Component*

O aspecto final do Molde será o seguinte:

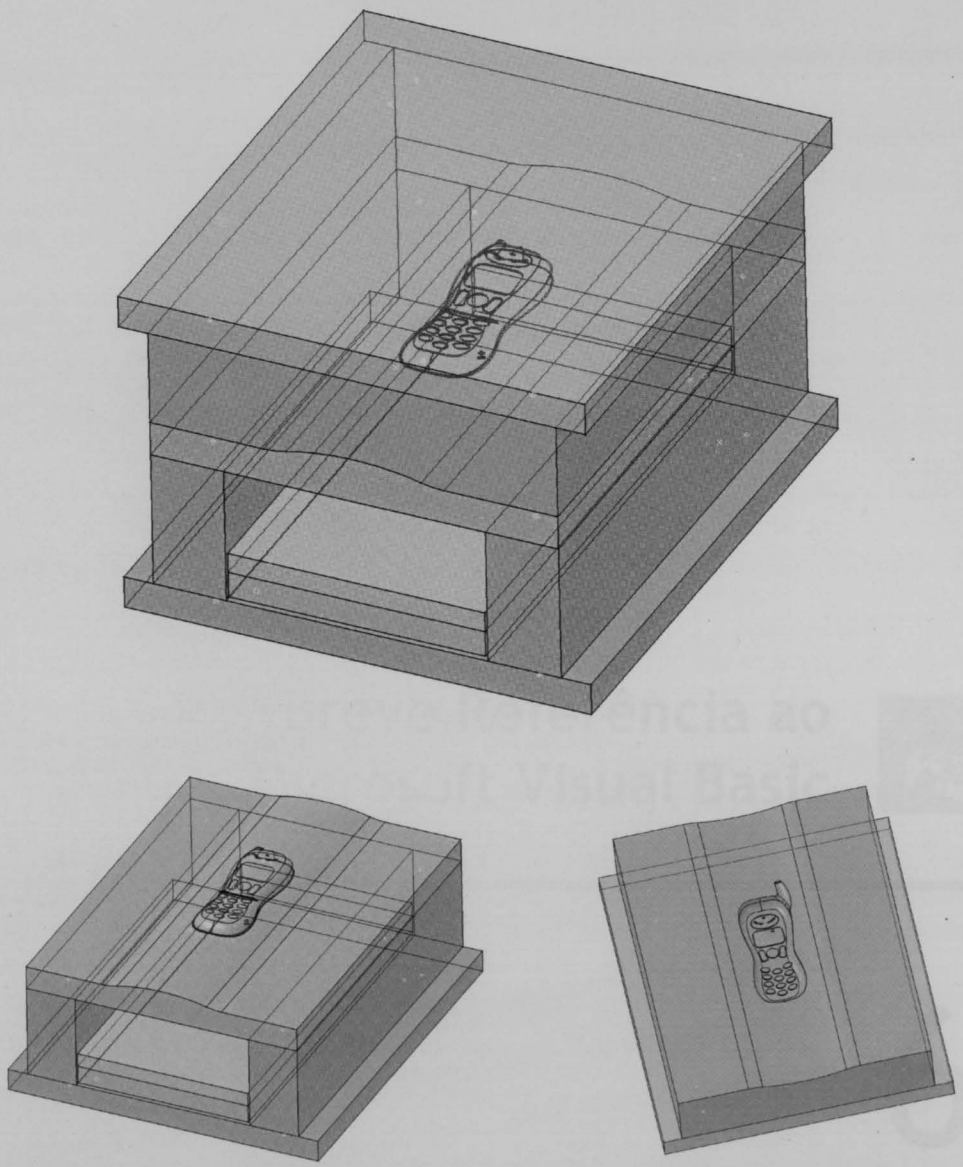


Fig. 30 – Aspecto final da Estrutura do Molde

Breve Referência ao Microsoft Visual Basic

6

6. Breve Referência ao Microsoft Visual Basic

O tipo de código que deu origem ao Visual Basic foi desenvolvido em 1963, dando pelo nome de **BASIC** (**B**eguinners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode). Este código tinha como grande finalidade, ser um tipo de codificação que fosse fácil e acessível a utilizadores com pouca experiência em programação. O *Basic* era a linguagem fornecida em muitos dos primeiros micro-computadores que começaram a aparecer nos anos 70. Este código continuou em desenvolvimento até o aparecimento do *Windows*. Em 1991 surgiu a primeira versão do Visual Basic, à qual seguiram outras versões espaçadas de um ano.

Quem teve a ideia de desenvolver o Visual Basic terá olhado à sua volta e percebido a filosofia da organização natural do mundo; o ambiente compõe-se de objectos, naturais ou artificiais, mas todos eles podem chamar-se objectos. Cada um destes objectos possui algumas propriedades e a cada um deles podemos atribuir outras (pesado/leve, amarelo/verde/vermelho, pequeno/grande, etc.), assim como podemos modificar algumas das características destas propriedades. Por fim, podemos actuar sobre os objectos, ou fazer com que eles próprios actuem (por exemplo, ao dar uma tacada numa bola de bilhar, ela vai bater numa segunda que, por sua vez, pode bater numa terceira, a qual cai num cesto).

Os fundamentos do Visual Basic são exactamente os mesmos. Temos uma interface gráfica (o ambiente), que é composta por objectos (formas e controlos); Cada objecto possui propriedades, sendo possível modificá-las alterando os seus atributos. Pode actuar-se também, sobre os objectos ou determinar actuações dos mesmos, atribuindo-se eventos a esses objectos através de códigos.

Portanto, o Visual Basic não obriga o programador a um exercício de raciocínio abstracto. Ao basear-se em objectos, tal como no mundo real, esta linguagem torna a tarefa de programar, de criar aplicações, relativamente mais fácil, natural e muito mais atractiva do que as outras linguagens existentes.

Para poder entender esta linguagem, vamos começar por definir alguns parâmetros fundamentais utilizados na codificação, entre os quais os "**Objectos**", as "**Propriedades**", os "**Métodos**" e os "**Eventos**".

Os objectos que nos rodeiam diferenciam-se pela sua forma física e química, ou seja, pelas suas propriedades. São estas que caracterizam os objectos. Em Visual Basic, como no universo, os objectos também têm propriedades. As propriedades podem ser alteradas, não no essencial porque os objectos não podem deixar de ser o que são, mas na quantificação. Podemos comparar com um copo de água; se aquecermos a água, esta não deixa de ser água, apenas alteramos a propriedade temperatura. Existem várias as propriedades que definem a água. Assim, também o Visual Basic cada controlo tem várias propriedades, as quais podemos ou não alterar, mas no entanto, o controlo nunca deixará de ser aquilo para que foi criado, apenas o moldamos, adaptando-o às nossas necessidades.

Na lógica do Visual Basic, um Objecto é um elemento visível numa aplicação. Por exemplo, um botão inserido numa Forma (*Form* - objecto que serve de base à interface - figura 31), ou uma imagem colocada na Forma, etc., tudo isto são objectos.

As propriedades serão a segunda parte da criação de um programa. Cada objecto é definido por um conjunto de características às quais podem ser atribuídos valores diferentes. Em VB chama-se propriedades as características de um objecto, e atributos aos valores dessas propriedades, ou seja, a propriedade *Height* (Altura), por exemplo, terá como atributo um número que determina a altura do objecto respectivo.

Um Método é uma função existente no Visual Basic que leva o objecto a executar uma Acção.

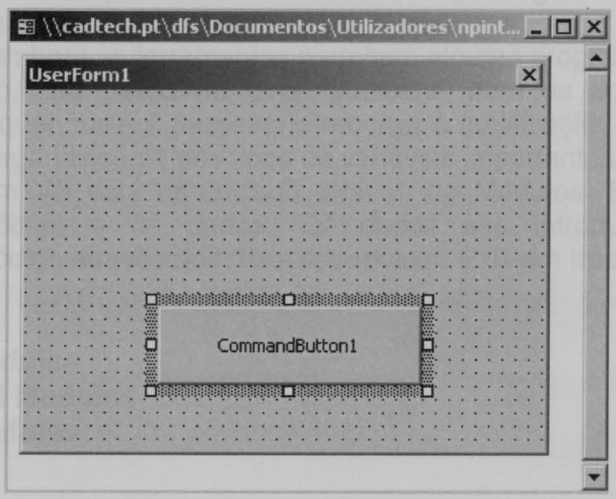


Fig. 31 – Objecto *Form* do Visual Basic

No mundo real, quando se abre uma torneira sai água da mesma. O acto de abrir a torneira é um "Evento". Como também é um Evento a inserção de uma moeda numa máquina de bebidas.

Como se pode perceber facilmente, qualquer destes eventos dá origem a uma resposta: a saída de água da torneira; a entrega de uma lata de bebida. Quer a torneira, quer a máquina, contêm instruções mecânicas de resposta a eventos de "abrir" e "inserir". Instruções que foram previamente introduzidas por quem as construiu e que resultam sempre na mesma resposta para cada um desses eventos. Em VB, passa-se tudo exactamente do mesmo modo: um Evento é essencialmente uma resposta de um objecto a uma ordem ou acção do utilizador, a partir de código escrito previamente que constitui o procedimento (Procedure).

Para exemplificar, podemos por exemplo, imaginar que temos o evento de dar um click um botão. A este evento responde uma acção que é desencadeada pelo código (ou Procedimento) que foi atribuído a esse objecto (figura 32).

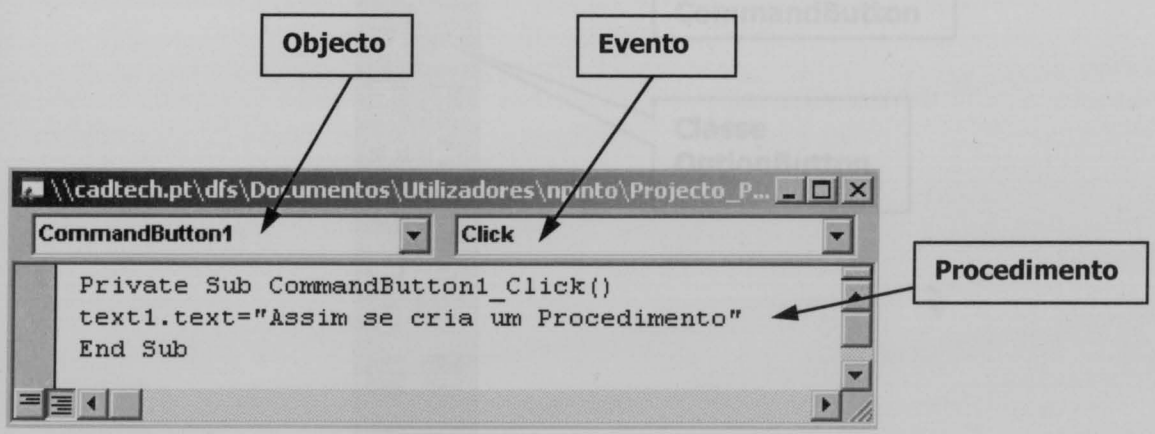


Fig. 32 – Janela de código indicando o Objecto, o Evento e o Procedimento.

Por ser uma linguagem visual, grande parte de uma aplicação em VB é programada visualmente, utilizando as ferramentas à disposição do utilizador. Estas ferramentas são utilizadas para construir a interface do programa, através da qual o utilizador vai interagir com o computador.

Contudo, e embora as ferramentas permitam alguma programação intrínseca, a interface, por si só, não passa de uma "Máscara" para as acções. Para que a aplicação faça qualquer coisa é necessário recorrer à codificação. De seguida vamos falar um pouco da organização e dos tipos de variáveis importantes.

O código em VB está organizado através de Módulos. Estes Módulos são contentores do código e de dados. Os dados são introduzidos através de "declarações" e o código através dos "Procedimentos". Existem três tipos de Módulos:

- ⇨ **Form Modules**
- ⇨ **Standard Modules**
- ⇨ **Class Modules**

Os *Form Modules* são os módulos que guardam o código referente aos objectos que são contidos no *Form*. Ao iniciarmos um novo projecto, nos é apresentado um *Form* e a este está associado o respectivo módulo.

Os *Standard Modules* são criados quando se pretende escrever código que, por não ser específico de nenhum controlo, deve estar à parte. Pode também acontecer que existam respostas iguais para eventos diferentes. Neste caso também se deve utilizar um módulo deste tipo para melhorar a estrutura do código.

Os *Class Modules* são utilizados quando se pretende criar novas classes. As classes são modelos de onde surgem os objectos. Cada objecto é uma cópia exacta do seu modelo. Como exemplo, podemos ver a figura 33 que nos apresenta as classes de objectos dentro do VB; são destas classes que surgem os objectos.

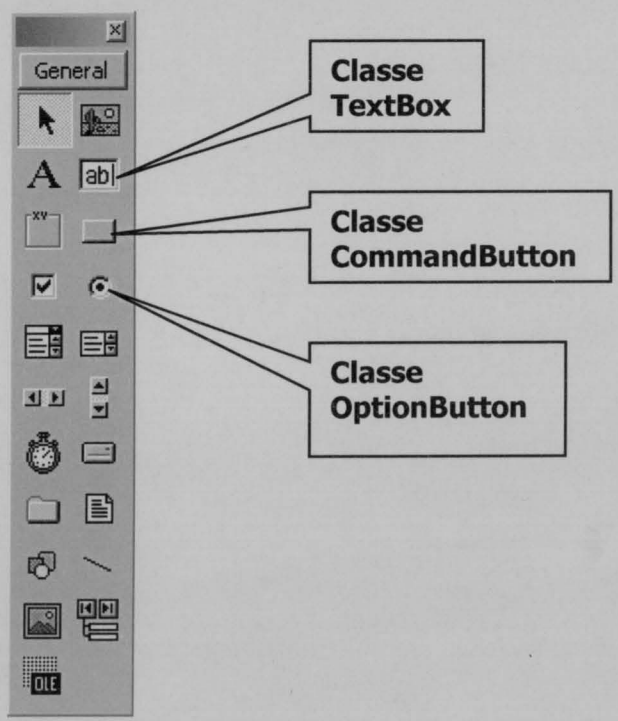


Fig. 33 – Caixa de ferramentas com as respectivas classes

Os objectos pertencentes às classes, geradas através deste tipo de módulos, são invisíveis, isto é, não podem ser colocados na caixa de ferramentas do VB.

Como já foi dito anteriormente, um módulo pode possuir procedimentos e dados. Os procedimentos são conjuntos de instruções que permitem trabalhar sobre os dados de modo a que possa produzir algum resultado. Os dados são guardados em variáveis e constantes.

Uma variável é uma localização em memória, utilizada para guardar informação durante a execução do programa. Para facilidade de programação, à variável é atribuído um nome. As variáveis são sempre de um determinado tipo. É este tipo que permite ao computador saber quais são os valores que a variável pode guardar.

O Visual Basic possui vários tipos de dados, entre os quais:

Tipo de dados	Valores
Byte	0 a 255
Boolean	True ou False
Integer	-32.768 a 32.767
Long	-2147483648 até 2147483648
String	0 até 2 biliões (aprox.)
Single	-3.402823 E 38 até 3.402823 E 38
Double	-1.79769313 E 308 até 1.79769313 E 308
Date / Time	1 de Janeiro de 100 até 31 de dezembro de 9999

Tabela 1 – Tipos de variáveis

Relativamente aos tipos de variáveis, será importante referir o tipo ***Variant***, visto ser um tipo importante tanto em VB como em VBA (no capítulo seguinte será explicado o conflito que existe entre os tipos de variáveis quando utilizamos a API (Application Programing Interface) do CATIA V5).

O Visual Basic permite a utilização de variáveis não declaradas. Estas variáveis possuem o tipo implícito ***Variant***. O *Variant* engloba todos os outros tipos e uma variável deste tipo pode conter valores inteiros, texto, datas, etc. Embora por vezes seja inevitável a utilização deste tipo de variáveis, a sua utilização extensiva não é recomendável, dado que conduz a uma programação pouco ordenada e a um desperdício de recursos (ocupam mais espaço em memória).

No capítulo seguinte, e depois de definidos os conceitos fundamentais, vamos dar início à abordagem da API (Application Programing Interface) do CATIA V5 para poder verificar as semelhanças e diferenças do **Visual Basic for Applications**, relativamente ao **Visual Basic**.

Aprendizagem da API do CATIA V5

7

7. Aprendizagem da API do CATIA V5

7.1 Documentação da API do CATIA V5

Em anexo ao programa principal, existem vários cd´s com toda a documentação de ajuda respectiva para todos os temas. Aqui podemos encontrar a ajuda referente a API de Automação. A leitura destes textos, constituiu a fase preliminar de aprendizagem dos Objectos, Propriedades e Métodos que constituem toda a API dedicada ao CATIA V5. Esta documentação está apresentada em formato HTML. Quando entramos nesta documentação, é apresentado um quadro (figura 34) que nos dá uma ideia da ligação entre as várias *Workbenches*, utilizando os objectos de Automação.

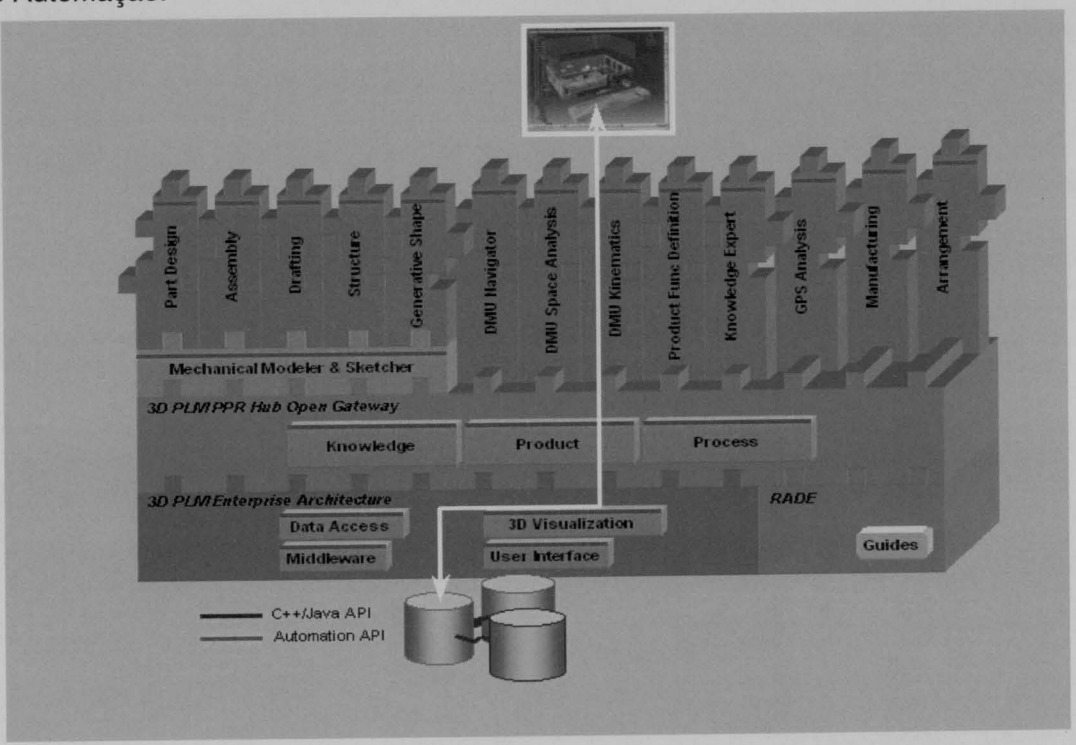


Fig. 34 – Quadro exemplificativo da arquitectura da documentação

Como podemos verificar, existem vários grupos divididos em várias cores. Do grupo que está a azul claro, podemos destacar o subgrupo mais importante que corresponde as *Workbenches PartDesign*, *Assembly*, *Drafting*, *Structure* e *Generative Shape*. Neste ponto podemos verificar que este grupo tem ligação com o grupo *Mechanical Modeler & Sketcher*, que por sua vez faz a ponte entre as estruturas de *Knowledge*, *Product* e *Process* (englobados no grupo verde). Esta arquitectura estará também ligada ao *3D PLM Enterprise Architecture* que engloba o *Data Access*, *3D Visualization*, *User Interface* e *Middleware*. Esta ligação entre todos estes grupos é importante para nos servir de guia quando acedemos a esta documentação, pois podemos limitar-nos a consultar as referências de que vamos necessitar. No nosso caso, estes vão ser os grupos mais importantes e aos quais foi dedicado mais atenção. Ainda, dentro de cada um destes grupos podemos encontrar o manual de Referências, onde podemos encontrar todos os objectos de todos os grupos e dentro destes, todas as propriedades e todos os métodos de cada objecto referenciado.

Para além de toda esta Documentação disponível, outra referência na aprendizagem da API foi a participação em *Forums* de CATIA V5, onde podemos encontrar uma secção exclusiva para quem se dedica ao desenvolvimento deste tipo de aplicações em todo o mundo. É necessário realçar que a participação nestes *Forums* foi fundamental, visto que existe uma troca de ideias muito útil com outros utilizadores.

Um dos *Forums* utilizados foi o **COE** (CATIA Operator's Exchange). Durante quase 20 anos, COE foi a única organização internacional, profissional que une e que suporta os utilizadores do software da Dassault Systèmes, que inclui o CATIA, ENOVIA, DELMIA e SMARTEAM, que conta com mais de 50.000 utilizadores em mais de 400 companhias desde o sector da Aeronáutica, Aeroespaço, Indústria Automóvel, etc. Este fórum contempla os seguintes *Forums*:

- General Discussions
- How is COE Doing?
- CADAM / Helix / CCD
- MFG
- Data Exchange
- V4 -> V5 Migration
- CATIA V5
- TeamPDM
- **CATIA V5 Programming**
- ENOVIA
- AEC & Shipbuilding
- RUGs (Regional User Groups)
- QA / Inspection

www.coe.org

Outro site disponível na internet com a mesma finalidade é o **CAAV5 Community**, sendo este um site que é propriedade da Dassault Systèmes. Neste caso é o próprio fabricante do software que nos permite aceder a um Fórum exclusivo para quem desenvolve aplicações em *Automation*, C++ & Java e XML. Este site também nos dá acesso a toda a documentação *on-line*.

CAA V5 Community
Share the power of the PLM Architecture

www.caav5.com

7.2 Curso de Formação: CATIA V5 Automation

Para além da leitura da documentação disponível, tive a oportunidade de participar num curso de formação em ***CATIA V5 Automation***, na IBM em Madrid com a duração de três dias. Este curso é dedicado a pessoas que trabalham no desenvolvimento de aplicações e como requisitos mínimos, deveriam ter experiência com CATIA V5 e ter conhecimentos de Visual Basic. Este curso foi dado pela Dassault Data Services, que faz parte da Dassault Systèmes.

CONTEÚDO DO CURSO:

✓ **CATIA V5 Automation & Scripting Architecture**

Apresentação

1. Diferenças entre VBScript, VBA e Visual Basic
2. Linguagens Suportadas
3. Arquitectura e Documentação

Running IN-Process Macros

1. CATIA V5 Visual Basic Editor
2. Livrarias de Macros
3. Gravar uma Macro
4. Correr uma Macro
5. Execução de uma Macro a partir de outra Macro
6. Execução de uma Macro com parâmetros
7. Adicionar uma Macro a uma Toolbar
8. Interface de Opções das Macros

Running OUT-Process Programs

6. Executar OUT- Process a partir de VBA ou Visual Basic
7. Exemplo: "Bolt from Excel"
8. Executando OUT- Process utilizando o Script Hosting do Windows
9. Executando OUT- Process em HTML

✓ **Scripting com CATIA V5**

1. Introdução
2. Scripting Infrastructure features
3. Scripting Sketches
4. Scripting Part Design
5. Scripting Shape Design
6. Scripting Assembly Design
7. Scripting Drafting
8. Automatic Selections
9. Graphic selections
10. Domínios de Aplicação
    - Análise
    - Manufacturing

11. Método de Programação
    - Numbers, Literals e Units
    - SafeArray Variant
    - A utilização do "Set"
    - A utilização do "Call"
    - Encriptar VBScript Macros
12. Administração
    - Registar CATIA e livrarias de Tipos
    - Converter uma Macro para um programa de Visual Basic
    - Converter um programa de Visual Basic para uma Macro
    - Macros em UNIX / NT

✓ **Como chamar outro Automation Server**

1. Chamar o WSH a partir do VBScript
2. Chamar o Word
3. Chamar o Excel
4. Criar um gráfico em Excel

✓ **Duração**

3 dias.

7.3 API do CATIA V5

Em primeiro lugar, vamos começar por definir as diferenças entre **VBScript**, **VBA** (Visual Basic for Applications) e **Visual Basic**.

O Visual Basic, como vimos anteriormente, é a versão completa. Nesta aplicação podemos criar programas independentes, *ActiveX* e *Servers*. O VBA é uma subaplicação que está embebida dentro do CATIA V5. Podemos encontrar a mesma situação em programas como o *Word*, *Excel*, *AutoCad*, etc. Todos estes programas têm um editor de Visual Basic onde se podem gravar e editar *Macros*, ou ainda, realizar aplicações mais complexas. O VBScript (Visual Basic Script) não é mais do que uma subaplicação do Visual Basic que utiliza uma linguagem mais simples onde não existe o dimensionamento das variáveis para que lhe seja atribuído um tipo.

É aqui que reside a grande diferença entre o que se chama de *Macros* e de aplicação. Nas *Macros* utilizamos a codificação do VBScript e não temos acesso a ferramentas de interface com o utilizador, ou seja, estamos limitados a executar sempre a mesma operação sem que seja possível alterar qualquer passo. Por exemplo, podemos criar uma *Macro* para desenhar um parafuso; mas o desenho do parafuso será sempre o mesmo e não vamos poder alterar qualquer uma das operações que levaram a cabo a sua concepção. A obtenção de uma *Macro* parte de uma gravação. Esta gravação é efectuada durante a execução das operações que levam a obtenção da peça. Para este efeito temos ao nosso dispor o menu da figura 35.

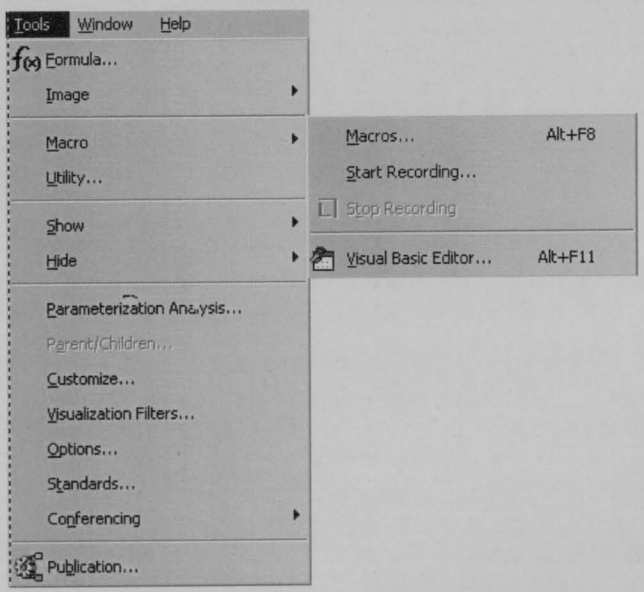


Fig. 35 – Menu onde podemos gravar e editar *Macros*

No caso de uma aplicação em VBA, temos ao nosso dispor todo o tipo de ferramentas de modelação de interface com o utilizador, podemos criar campos onde podem ser introduzidos parâmetros, podemos fazer com que o utilizador seleccione interactivamente um objecto que seja necessário à obtenção da peça final, etc. Como resultado, com um pouco de imaginação, podemos criar uma interface que seja agradável ao utilizador e lhe permita executar as operações de um modo fácil e mais rápido possível.

O CATIA comunica com outros processos, com a ajuda as Interfaces **COM** (Component Object Model), como descrito na figura 36.

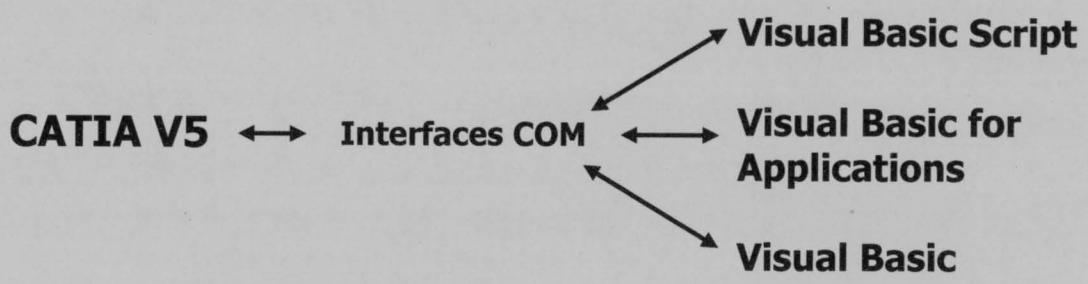


Fig. 36 – *Automation*: Comunicação entre vários Processos

Tendo estes conceitos definidos, podemos avançar na definição de ***In-Process Application*** e ***Out-Process Application***. No *In-Process Application*, utilizamos uma *Macro* que será produto de uma gravação, como referido anteriormente. Neste caso, quando a *Macro* é activada o CATIA é desactivado. Esta codificação é efectuada em VBScript, VBA em Windows e VBScript (Winsoft) em UNIX. Relativamente ao *Out-Process Application*, é um processo que é externo ao CATIA que o chama, cria ou modifica a geometria, etc. O CATIA não é desactivado. Este tipo de aplicações pode ser desenvolvido em Visual Basic, Windows Scripting Host (VBScript, Jscript), Windows Explorer HTML (VBScript, Jscript) e qualquer outra aplicação COM.

As linguagens como o Visual Basic, trabalham com Objectos (como vimos anteriormente). Todos estes objectos têm as suas propriedades. A forma de utilizar estes objectos dependem dos Métodos implementados. Para além disto, existem as Colecções que são listas de objectos que podem ser utilizados ao mesmo tempo. Resumindo (na API CATIA V5):

⇨ **Objecto:** é uma Entidade.

Ex: Um documento, um *Pad*, uma Linha, etc.

⇨ **Propriedade:** Característica de um Objecto.

Ex: NomeDocumento = CATIA.ActiveDocument.*FullName*

⇨ **Método:** é uma Acção realizada num Objecto.

Ex: CATIA.ActiveDocument.*SaveAs* "Novo Molde"

⇨ **Colecção:** lista de Objectos.

```
Ex: For i=0 to CATIA.Documents.Count
        MsgBox CATIA.Documents.Item(i).Name
    Next
```

Nota: é de realçar que quando utilizamos ciclos para percorrer todos os elementos que verificam uma condição, não podemos começar em i=0, mas sim em i=1.

No caso da **API** (Application Programing Interface) do CATIA V5, todos estes Objectos são entendidos como funções que existem dentro do próprio programa e onde estão disponíveis todos os comandos de todas as *Workbenches*, que podem ser utilizados desde que se siga todas as regras impostas pela própria dinâmica de declaração dos métodos que levam a que se obtenha a função desejada. Existe alguma diferença relativamente ao que foi dito no capítulo 6, visto que agora estamos a falar de objectos com um carácter gráfico e que dependem das funções de um programa base, que neste caso é o CATIA, e das funções que estão disponíveis dentro da API.

Por vezes existem conflitos entre os tipos de variáveis próprias do Visual Basic e o tipo de variáveis do CATIA. Esta situação pode levar a acontecerem erros quando estamos a correr a aplicação. Esta situação pode ser facilmente resolvida, optando nestes casos por declarar as variáveis como ***Variant*** em vez do tipo de variável correspondente.

Para ter alguma ajuda na pesquisa dos objectos e seus métodos, temos também ao nosso dispor, dentro do ambiente de trabalho do VBA, o ***Object Browser***. Esta ferramenta é extremamente útil para pesquisar toda a API introduzindo apenas uma palavra chave, como podemos verificar na figura 37.

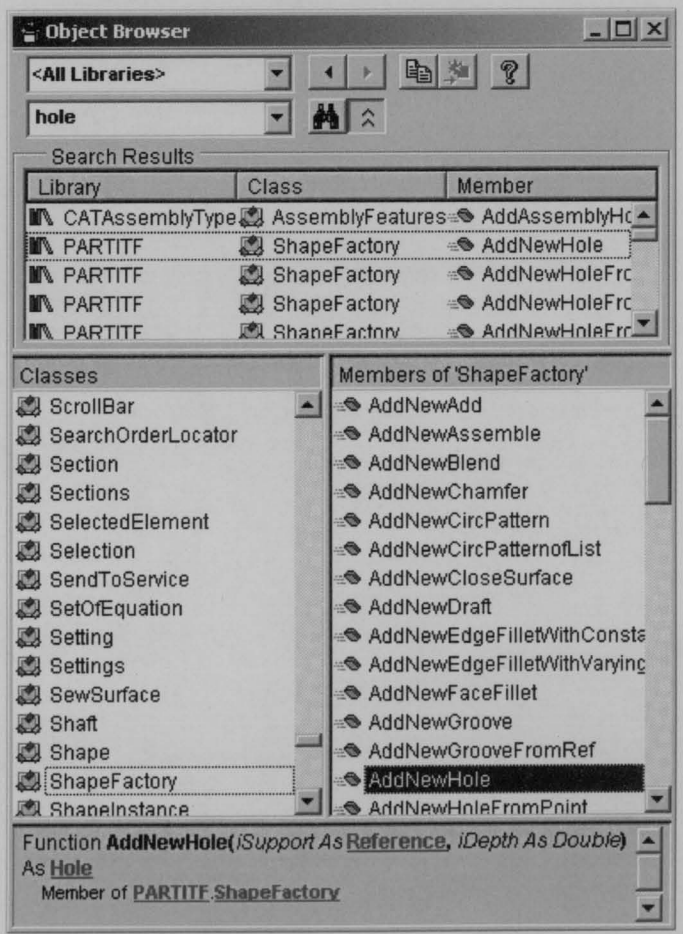
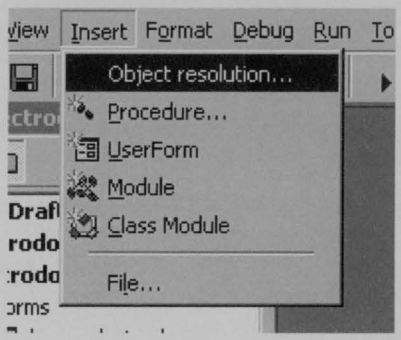
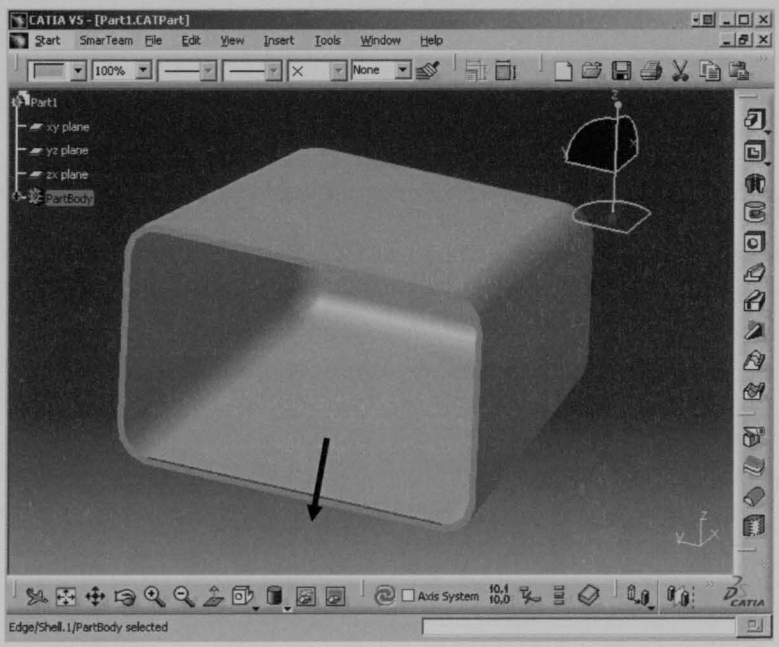


Fig. 37 – *Object Browser*: Pesquisa através de uma palavra chave, por exemplo: *Hole*

Para finalizar podemos ainda falar no ***Object Resolution***. Esta é mais uma das ferramentas exclusivas do VBA do CATIA V5 que nos permite obter directamente as referências de um objecto gráfico do CATIA (por exemplo, uma face, uma aresta, um ponto, etc.). Por exemplo, se formos ao menu da figura 38 e clicarmos em *Object resolution*, podemos verificar que nos é apresentado o ambiente de trabalho do CATIA. Se seleccionarmos uma aresta podemos verificar que as referências deste objecto foram transcritas para o módulo em que estamos a trabalhar.





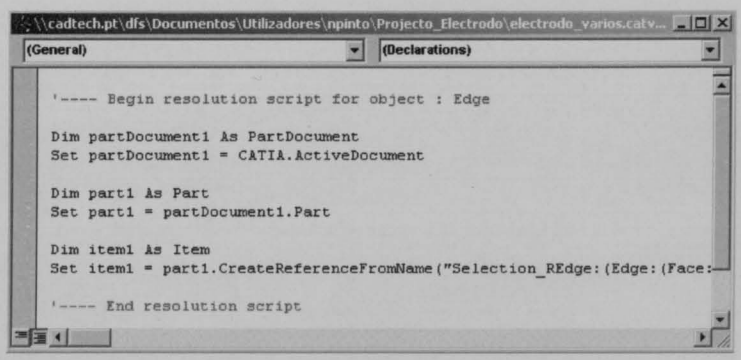


Fig. 38 – Procedimento para aceder ao *Object Resolution*

7.4 Exemplo de aplicação

Um dos exemplos seguidos para iniciar a aprendizagem da API do CATIA V5, foi a execução de uma aplicação que cria uma "Árvore de Cames" muito simplificada. Este exemplo está disponível na documentação na forma de VBScript. O objectivo era tentar pegar no VBScript e modelá-lo de maneira a obter uma aplicação que tivesse uma interface gráfica que exemplificasse qual era a sua função, e onde se pudesse modificar os parâmetros de forma a obter vários objectos com medidas diferentes.

Para este efeito, foi concebida primeiro a interface. Esta interface contém vários botões entre os quais um botão para "Criar" a peça, outro para "Limpar" todos os campos ao mesmo tempo, outro para assinalar a "Versão", outro para colocar nos campos os "Valores Predefinidos" e, por último, um botão para "Sair" da aplicação (figura 39).

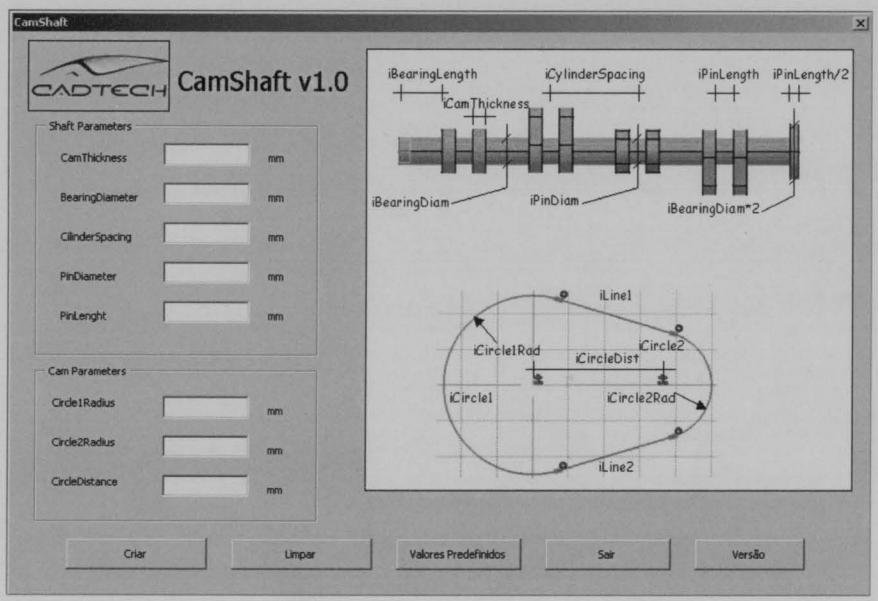


Fig. 39 – Interface do *CamShaft*

Os campos correspondem a medidas do *Sketch* representado na figura inicial e às medidas correspondentes ao Veio, nomeadamente as distâncias entre Cames e as espessuras. A metodologia baseia-se na obtenção de um *Sketch*, no qual se coloca cada entidade nas coordenadas correctas de modo a formar um perfil fechado para que depois sejam atribuídas as respectivas restrições (figura 40).

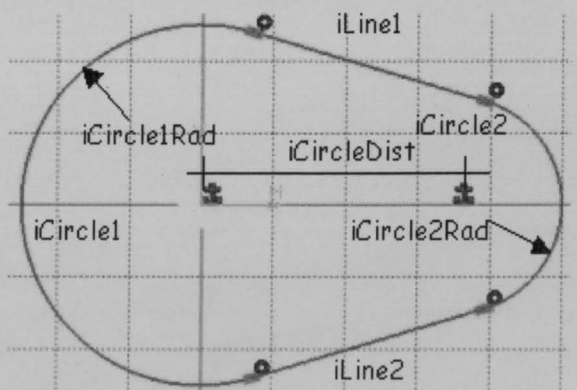


Fig. 40 – *Sketch* necessário à obtenção de um Came

A partir daqui, começa a concepção em 3D resultando em operações repetidas, mudando apenas um parâmetro que será definido como sendo a cota que o *Sketch* avança em **xx**, visto que a sua base está assente no plano **zy** (figura 41).

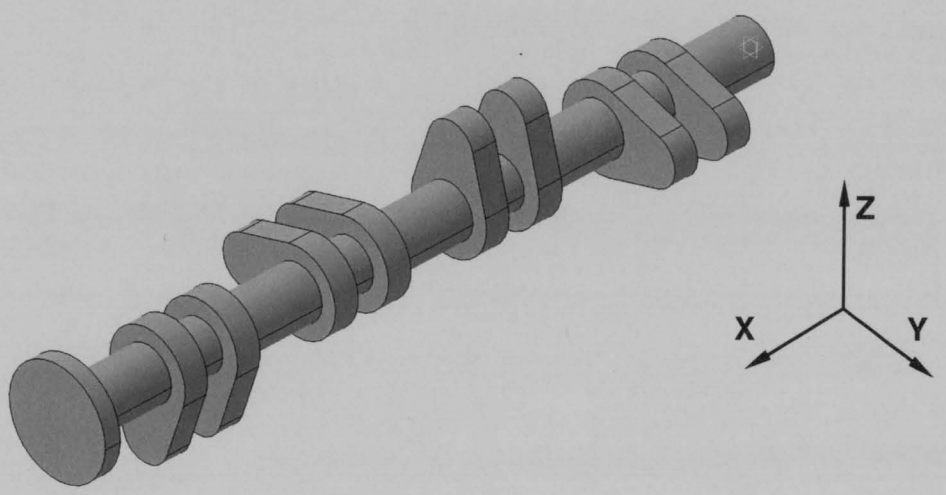


Fig. 41 – Base do *Sketch* assente no plano zy, propagação da criação segundo xx

Estas operações repetidas são resultado da utilização de Módulos que contêm dentro de si, a parte do código necessária para a obtenção de uma parte da peça, que chamado várias vezes, mudando apenas os parâmetros referidos atrás, desenvolve a peça. Como resultado, obtemos a peça pretendida e com as dimensões que foram introduzidas.

Depois deste exemplo de aplicação, vamos dar início ao desenvolvimento das nossas aplicações.

Concepção de uma Ferramenta para a Execução de Eléctrodos para o Processo de Electroerosão

8

8. Concepção de uma Ferramenta para a Execução de Eléctrodos para o Processo de Electroerosão por Penetração

8.1 Visita à TJ Moldes, S. A.

Uma das etapas mais importantes para a execução deste Projecto de Fim de Curso, foi a visita a duas das empresas clientes da CadTech. Como não poderia deixar de ser, existiu a necessidade de numa fase inicial, sondar quais eram as necessidades da Indústria de Moldes, tendo como referência um dos mais prestigiados construtores de Moldes da zona da Marinha Grande, a **TJ Moldes**.

O ramo de actividades da TJ Moldes é o fabrico de moldes em aço para a indústria de plásticos. Com um capital social de 500 mil contos é, desde 1990, sociedade anónima e participa no capital de mais três fábricas (duas de moldes e uma de plásticos). As empresas estão todas localizadas na Marinha Grande num polo industrial que a TJ Moldes criou e onde trabalham cerca de 130 trabalhadores. Em 1998 facturou cerca de 1.600 mil contos. A TJ Moldes tem a capacidade produtiva dividida por várias unidades, podendo assim oferecer ao cliente o rigor da qualidade em função da dimensão e do produto desejado.

PRODUTOS

- ⇨ Moldes pequenos até 2.000 kg
- ⇨ Moldes médios até 10.000 kg
- ⇨ Moldes grandes até 30.000 kg
- ⇨ *Try-outs* até 1.300 ton.

8.2 Necessidades da Empresa

Na primeira visita a empresa em questão, existiu a tentativa de perceber quais seriam as tarefas que, na execução de um Molde, poderiam ser agrupadas para que a fase de desenho fosse o mais eficiente possível. Para isto, foi necessário estar perto das pessoas que trabalham com CATIA V5 na secção de desenho.

Em primeiro lugar, é importante realçar que os Moldistas por vezes não utilizam na totalidade todas as capacidades do software. A passagem de utilizadores de CATIA V4 (que era um software que funcionava numa estação UNIX) para CATIA V5 (sendo este um software que funciona num PC em ambiente Windows) revelou-se difícil, visto que, existiu uma mudança muito grande entre as duas versões. Em quanto que com CATIA V4 tínhamos um software que era excelente para modelação de superfícies, mas limitado na modelação dos sólidos, com o aparecimento do CATIA V5 esta situação foi ultrapassada. Funcionando num PC, que por si só já é uma vantagem, visto que antes era necessário ter um posto de trabalho completamente dedicado ao CATIA V4, o CATIA V5 surge com uma metodologia completamente nova, ao ponto de se tornar um produto inovador. É precisamente por esta razão que a migração se torna difícil; por vezes existe a tentativa de trabalhar com CATIA V5 da mesma forma que se trabalhava com CATIA V4, o que se traduz na obtenção de alguns erros principalmente quando o desenho se torna complexo, como no caso dos Moldes. Aqui reside a razão pela qual no início deste texto, foram insistentemente introduzidos conceitos fundamentais para trabalhar correctamente com CATIA V5, e que conseguem fazer com se perceba a lógica de concepção da ferramenta em causa. A concepção destas ferramentas, não se destina a suprimir erros do utilizador, muito pelo contrário, só funcionará com a utilização de uma metodologia correcta.

A primeira ideia que surgiu, foi a concepção de uma ferramenta para realizar Eléctrodos para o processo de Electroerosão a partir da superfície de separação do Macho / Cavidade. Esta ferramenta deveria suprimir as tarefas necessárias à obtenção do Eléctrodo, e ainda tendo a vantagem de obrigar à utilização da metodologia correcta. Esta ferramenta tinha a particularidade de ter a necessidade de trabalhar em separado do resto do desenho do molde, ou seja, após a obtenção das superfícies resultantes da execução da separação Macho / Cavidade (explicada anteriormente), só será necessário ter uma cópia do ficheiro da *Molded Part* em separado do Molde, visto já que a pessoa que desenha o Molde não é a mesma que executa a separação Macho / Cavidade e os eléctrodos. Para além disto, é necessário que exista um ficheiro, fisicamente separado para cada eléctrodo para que sejam enviados para o software de CAD/CAM existente nesta empresa. Esta independência entre os trabalhos tem vantagens em termos de tempo, mas tem desvantagens em termos de CAD, visto que com esta metodologia existe a perda do *Link* entre a *Molded Part* do Molde e a *Molded Part* para os Eléctrodos; se existir a necessidade de haver uma modificação na superfície após a execução dos Eléctrodos, o CAD perderá toda a sua regenerativídade, e não vai actualizar a superfície que faz os eléctrodos. Ainda, é necessário que os sólidos tenham uma base para que possam funcionar correctamente na máquina de Electroerosão. Também existe a necessidade que a aplicação faça o desenho rápido do Eléctrodo em 2D utilizando um formato de folha definido pela empresa. Esta é a metodologia de trabalho existente na TJ Moldes para a execução dos Eléctrodos e que nos propusemos seguir.

Na execução de um Molde, esta empresa trabalha com um sistema de eixos rígido, principalmente no que diz respeito ao posicionamento dos Postiços dentro da estrutura do Molde. Esta empresa faz Moldes de grandes dimensões, e por vezes trabalham sobre 2 ou 3 Postiços, que depois juntam ao projecto final aplicando o *Assenbly Design*.

Foi a partir daqui que surgiu a ideia de criar uma aplicação que fosse capaz de colocar os Postiços dentro do Molde, utilizando este sistema de eixos que é o mesmo para os postiços e para o Molde.

Outra ideia proposta, foi a concepção de uma estrutura *Standard* de um Molde definido para a empresa, visto que a TJ Moldes não compra a estrutura a nenhum fornecedor (como HASCO, DME, etc.), mas sim a realiza comprando as placas de aço e executando a sua maquinação, segundo as regras internas definidas pela própria empresa. Apesar de interessante esta ideia foi abandonada, visto que, para a concepção de uma estrutura como esta, era necessário quebrar certas regras impostas pela metodologia do MTD. Este é um problema que teria de ser pensado de uma forma diferente e devido à falta de tempo para a conclusão deste Projecto não será viável, pelo menos por enquanto.

Para se ter uma ideia da forma como a ferramenta pode ser personalizada ao cliente através de pormenores muito simples, podemos referir que na Marinha Grande a forma de desenhar os Moldes é diferente à forma de os desenhar na região Norte do país. Na sua concepção o Molde é virado ao contrário, pondo a parte móvel para cima e a parte fixa para baixo (figura 42); surgiu também a ideia de fazer uma aplicação muito simples que fosse capaz de inverter o sistema de eixos ao gosto do cliente, permitindo maior comodidade de utilização.

De todas estas ideias e de outras que foram surgindo, a opção recaiu em desenvolver uma ferramenta para conceber **Eléctrodos**, visto ser esta a opção de maior necessidade.

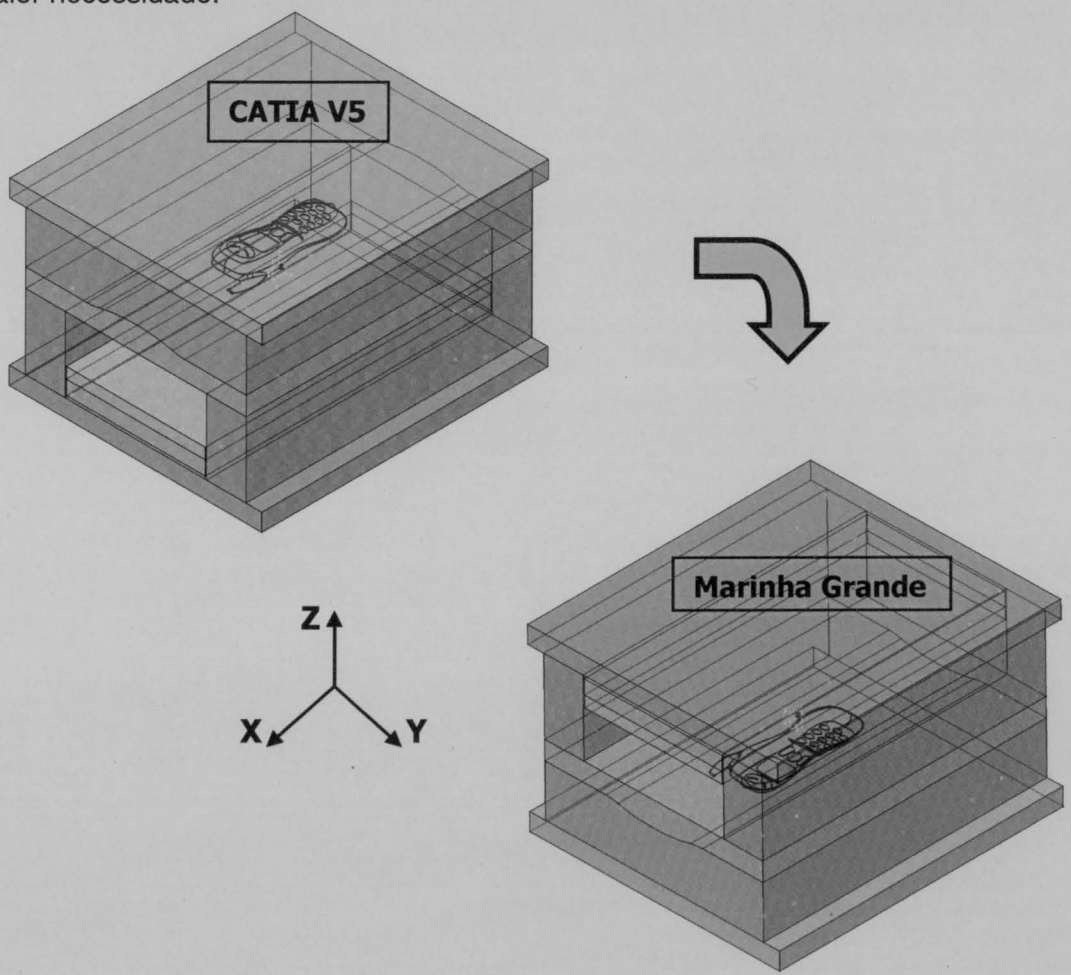


Fig. 42 – Sistema de Eixos do Molde utilizado na Marinha Grande

8.3 Concepção e Desenvolvimento da Aplicação

Para a concepção de uma ferramenta deste tipo, é necessário conhecer todas as etapas que levam à Modelação de um **Eléctrodo**. Para começar, é necessário que se obtenham as superfícies de partição, que surgem após a separação Macho / Cavidade. Estas superfícies estão todas incluídas dentro da *Molded Part* do nosso Molde. Estas superfícies vão ter o nome de *CavitySurface* e *CoreSide* (figura 43).

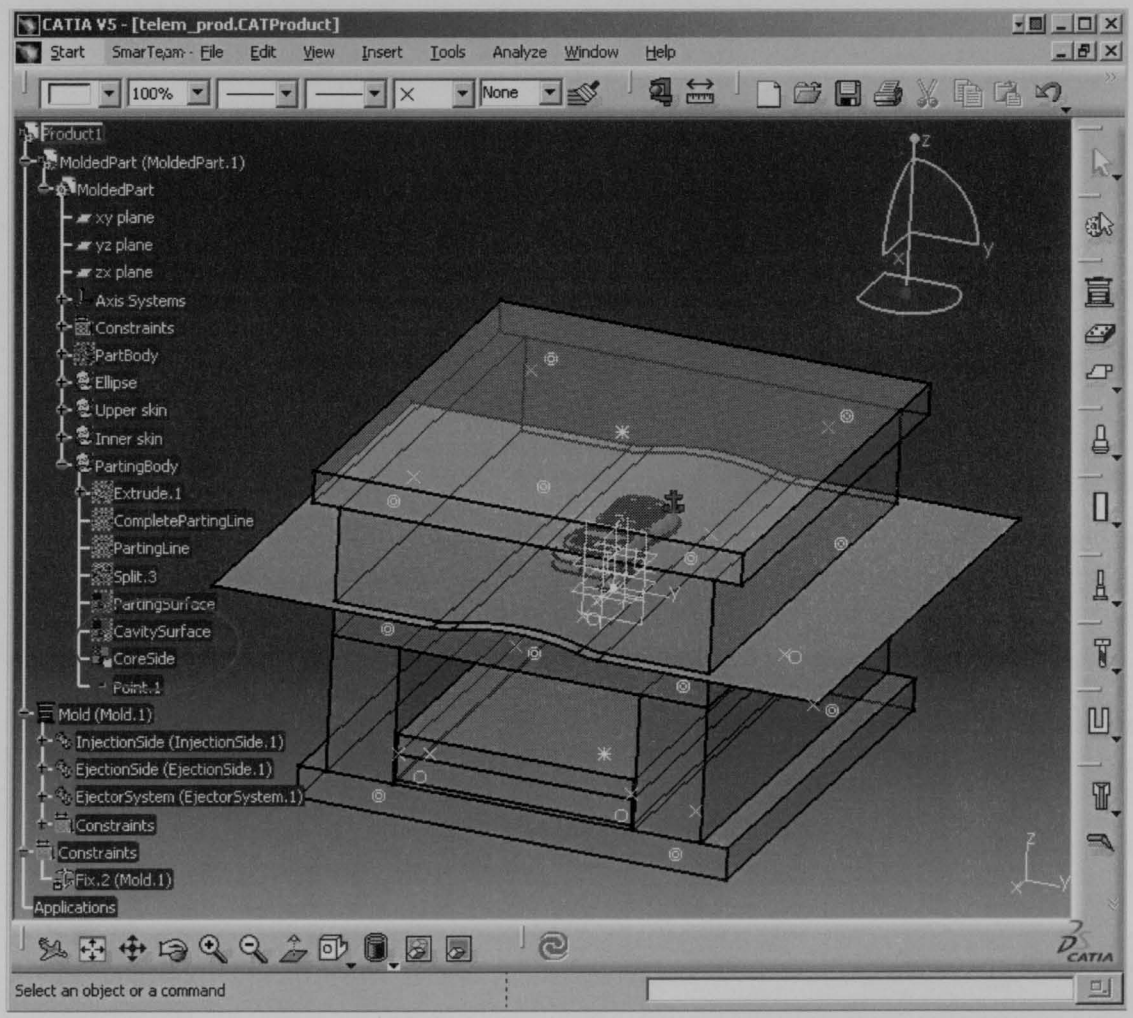


Fig. 43 – *Molded Part* necessária para criar o novo produto

O primeiro passo é criar um novo Produto. Depois é necessário adicionar a nossa *Molded Part* a este novo produto. Esta *Molded Part* fará parte de dois produtos, um como sendo o Molde e o outro como sendo o novo Produto para criar os Eléctrodos. Agora é necessário adicionar uma nova *Part* para cada Eléctrodo que se pretende criar (figura 44), com a finalidade de se ter um ficheiro para cada e que possa ser enviado em separado para a maquinação. A razão pela qual é necessário criar um Produto à parte, reside no facto de se fazer com que os sólidos pertencentes aos Eléctrodos, não façam parte do Molde apesar de serem obtidos a partir deste.

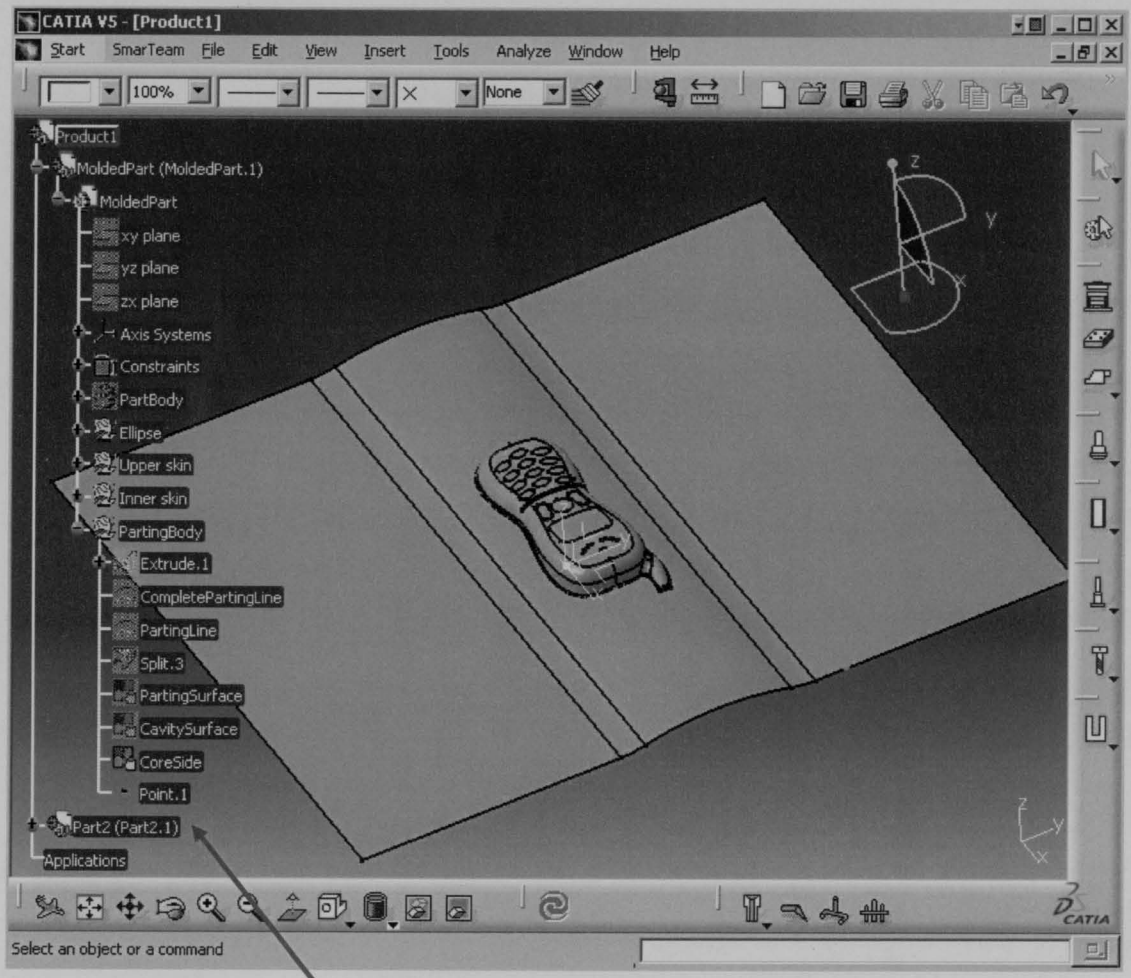


Fig. 44 – Adição de uma nova *Part* ao novo produto para a execução do novo Eléctrodo

O próximo passo será criar um *Sketch* com a secção pretendida do Eléctrodo (figura 45). Para isto, é necessário que o plano que vamos utilizar, esteja situado numa cota superior à da superfície em questão. Se necessário, podemos criar um plano auxiliar com um *offset* relativamente a um plano de referência, que neste caso será o plano **XY** (figura 46). Esta cota será necessária visto que, a execução do Eléctrodo será feita com a separação de um sólido, criado a partir do *Sketch*, pela superfície de partição respectiva (Macho ou Cavidade); se o sólido não atravessar por completo a superfície, corremos o risco de não conseguir criar o nosso Eléctrodo com a forma pretendida.

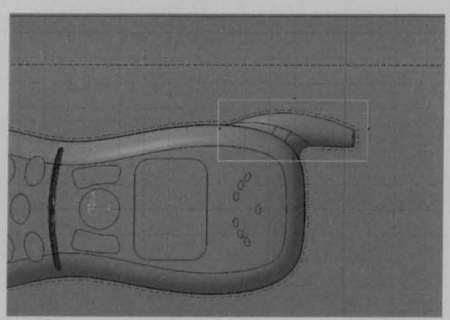


Fig. 45 – *Sketch* com a forma do Eléctrodo (rectângulo)

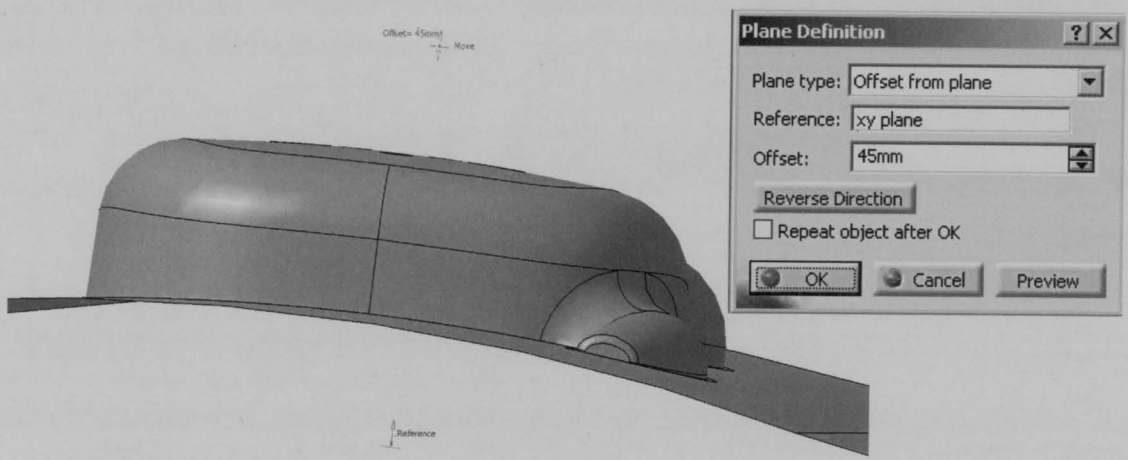


Fig. 46 – Criação de um Plano auxiliar (superfície do Macho)

No caso da Cavidade, não existirá este problema visto que os planos de referência se situam acima da superfície (figura 47).

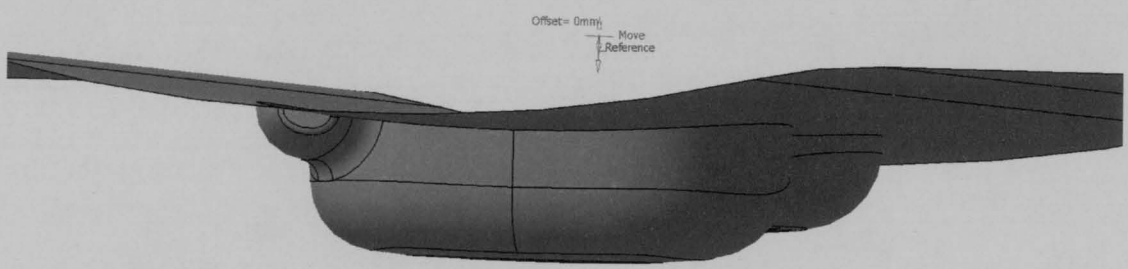


Fig. 47 – Superfície da Cavidade

O próximo passo é fazer um *Pad* com o *Sketch* que acabamos de criar dentro da nossa nova *Part*. O comprimento do *Pad* será definido pelo utilizador e deverá ser o suficiente para que a superfície corte o sólido obtido. A fase seguinte será efectuar a separação do sólido com a superfície, ficando sempre com um dos lados, que no caso dos Eléctrodos, será o lado de fora da placa (figura 48).

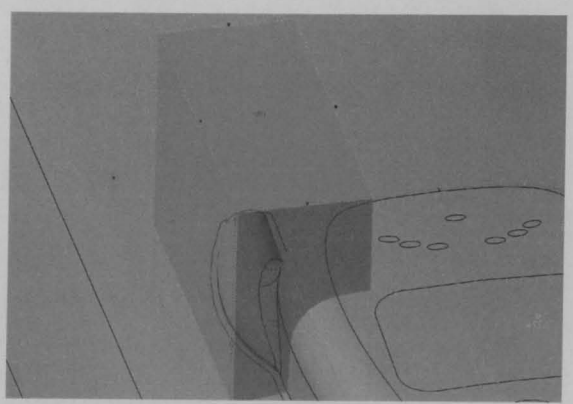


Fig. 48 – *Pad* para o Eléctrodo

Para isto, vamos utilizar uma função que existe no CATIA que se chama ***Split***, e à qual temos acesso sempre que entramos na *Workbench* de *Part Design*. Para executar o *Split* , vamos ter de indicar qual é a superfície com que vamos fazer o *Split*, e depois indicar com que lado pretendemos ficar (figura 49).

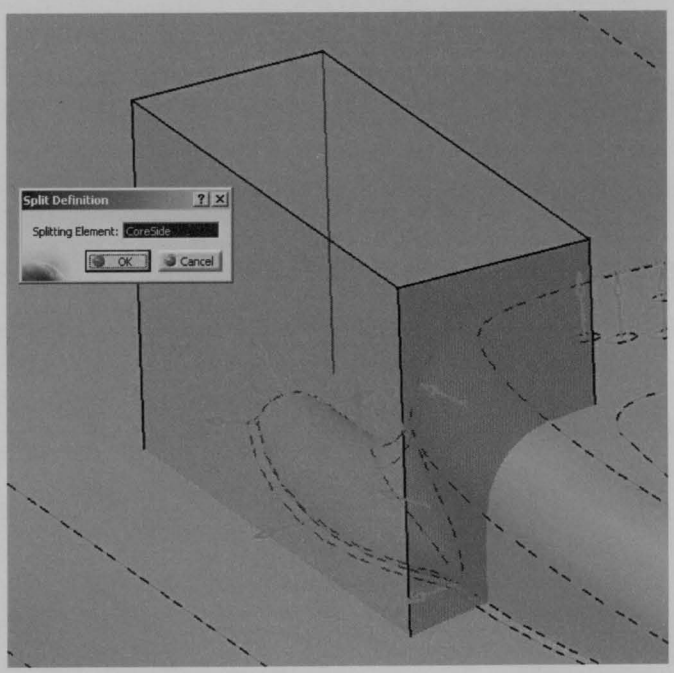


Fig. 49 – *Split* do Sólido do Eléctrodo

Resultando:

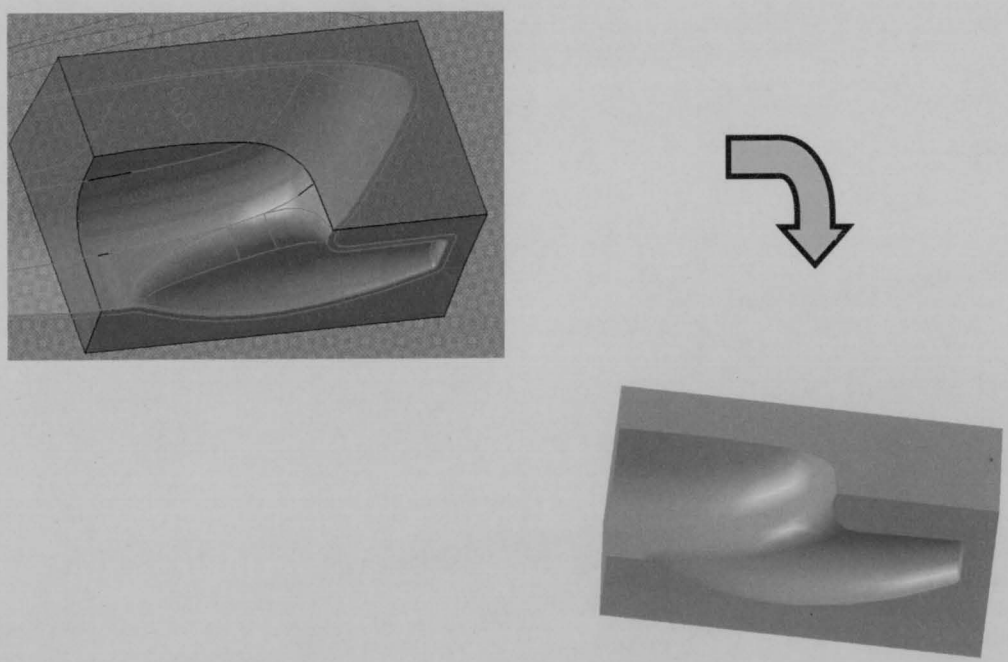


Fig. 50 – Forma do Modelo do Eléctrodo depois do *Split*

Após todas estas operações resulta o sólido pretendido e com a forma final do Eléctrodo faltando a sua base. A base é construída a partir de um *Offset* do *Sketch* inicial, no qual se executa um novo *Pad* com sentido contrário ao do inicial. Todas estas dimensões são introduzidas pelo utilizador. A forma final de um Eléctrodo será a da figura 51.

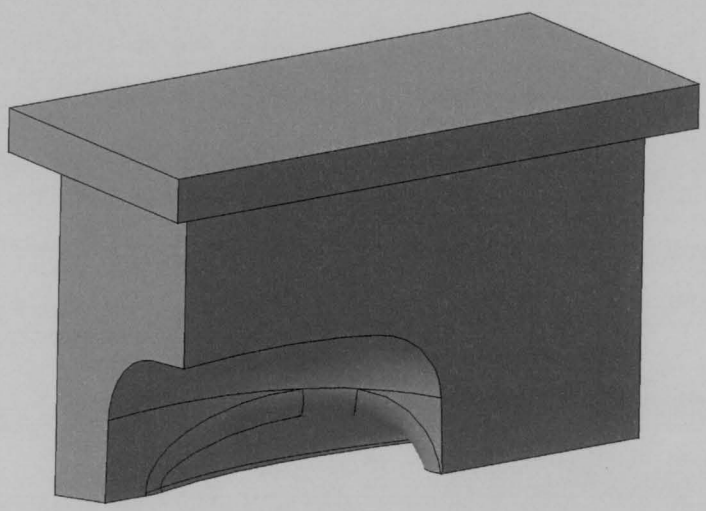


Fig. 51 – Modelo do Eléctrodo para o processo de Electroerosão da parte Macho

É necessário ter em consideração que, a definição do lado do Macho ou da Cavidade é muito importante, visto que, o sentido de execução do *Pad* será inverso ao do apresentado acima (figura 52). A utilização da representação da execução de um Eléctrodo no Macho foi feita apenas para poder representar a função do "Plano Auxiliar" e em que condições deve ser utilizado.

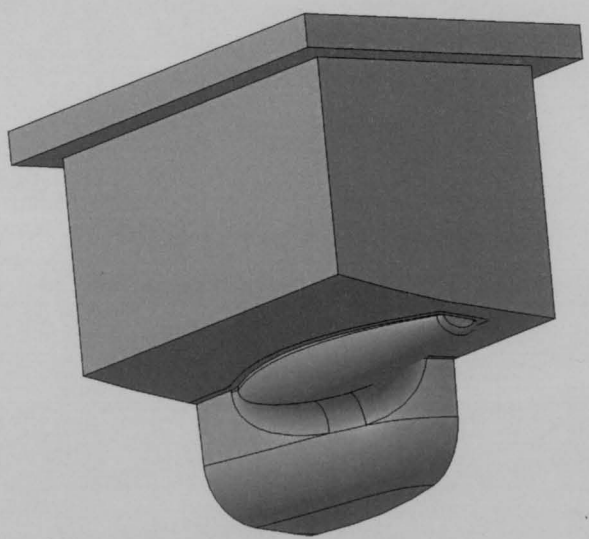


Fig. 52 – Modelo do Eléctrodo para o processo de Electroerosão da parte Cavidade

Ao executar esta série de operações, a estrutura da árvore da nova *Part* criada para o Eléctrodo terá a seguinte estrutura final:

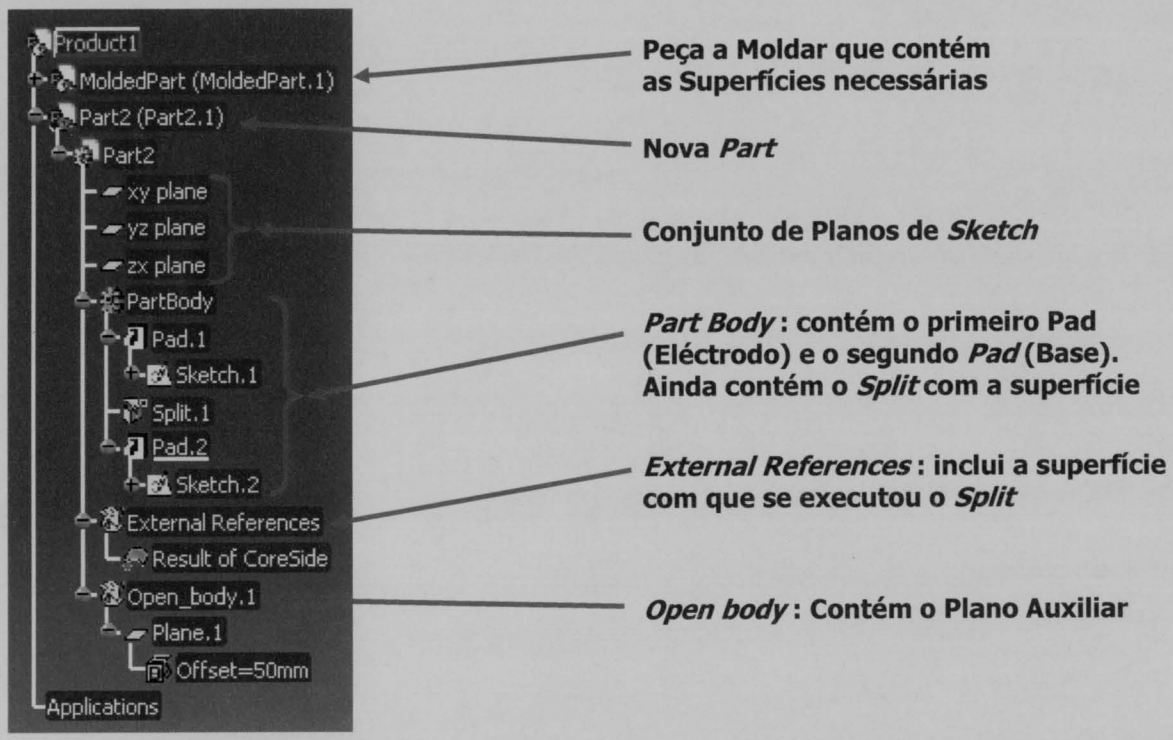


Fig. 53 – Estrutura da árvore do Produto para executar os Eléctrodos

Depois de definidas claramente todas as operações que levam à obtenção do Eléctrodo, estamos agora em condições de começar a definir a nossa aplicação.

O primeiro passo a seguir no desenvolvimento de qualquer aplicação, é definir a interface com o utilizador. Nesta interface, vamos ter de introduzir todos os campos, a preencher pelo utilizador, que são necessários a execução do Eléctrodo. Para além disto, é necessário que a interface defina com clareza quais são as operações que a aplicação executa e esteja embebida num ambiente agradável, tanto quanto possível.

Na interface desenvolvida, existirá um campo onde podemos introduzir o nome da *Part* a processar. Este nome deverá ser o nome do ficheiro que será gravado na pasta onde se encontra o nosso novo Produto. Visto que podemos querer mudar o nome da *Part* dentro do CATIA para, por exemplo "Electrodo 103", o nome que vai interessar para o processamento será o nome do ficheiro com a terminação "**.CATPart**". Esta terminação será a que vai determinar se é uma *Part* ou um *Product* e terá que ser preenchida correctamente.

Vamos ter também, um campo onde vamos introduzir a "**Cota do Sólido Total**". Este parâmetro vai ser o valor que vamos atribuir para o comprimento do *Pad*, que foi explicado anteriormente. Parece evidente que é necessário fornecer ao utilizador mais ajuda visual na escolha dos parâmetros para realizar os Eléctrodos. Para isto vamos colocar duas imagens na interface; a primeira será a imagem final de um Eléctrodo e a segunda, será uma imagem para exemplificar o significado do parâmetro da "Cota do Sólido Total" (figura 54).

Em termos de botões, vamos necessitar de um para dar início à criação do Eléctrodo (**1**). A este botão será atribuído a maior parte do código. Como é evidente, vamos necessitar de um botão para "Sair" da aplicação (**3**). Ainda vamos necessitar de mais três botões; um para assinalar a "Versão" da aplicação (**4**), o outro será um botão de "Ajuda" (**5**) onde vamos ter ligação a um ficheiro de texto que fará a explicação pormenorizada do funcionamento de toda a aplicação (explicação dos parâmetros, limitações, ponto de partida, exemplos de funcionamento, etc.), e por último, vamos ter a necessidade de ter um botão para "limpar" todos os campos ao mesmo tempo (este último, apenas por razões de comodidade) (**2**) (ver figura 55).

Após uma segunda visita à TJ Moldes, surgiu a ideia de colocar uma opção onde se pudesse dizer se se pretendia trabalhar com histórico ou não, ou seja, com ou sem *Link* (figura 54). A noção do *Link*, as suas vantagens e desvantagens, já foram explicadas anteriormente. No entanto, agora vamos poder ter a oportunidade de perceber melhor esta opção de trabalho.

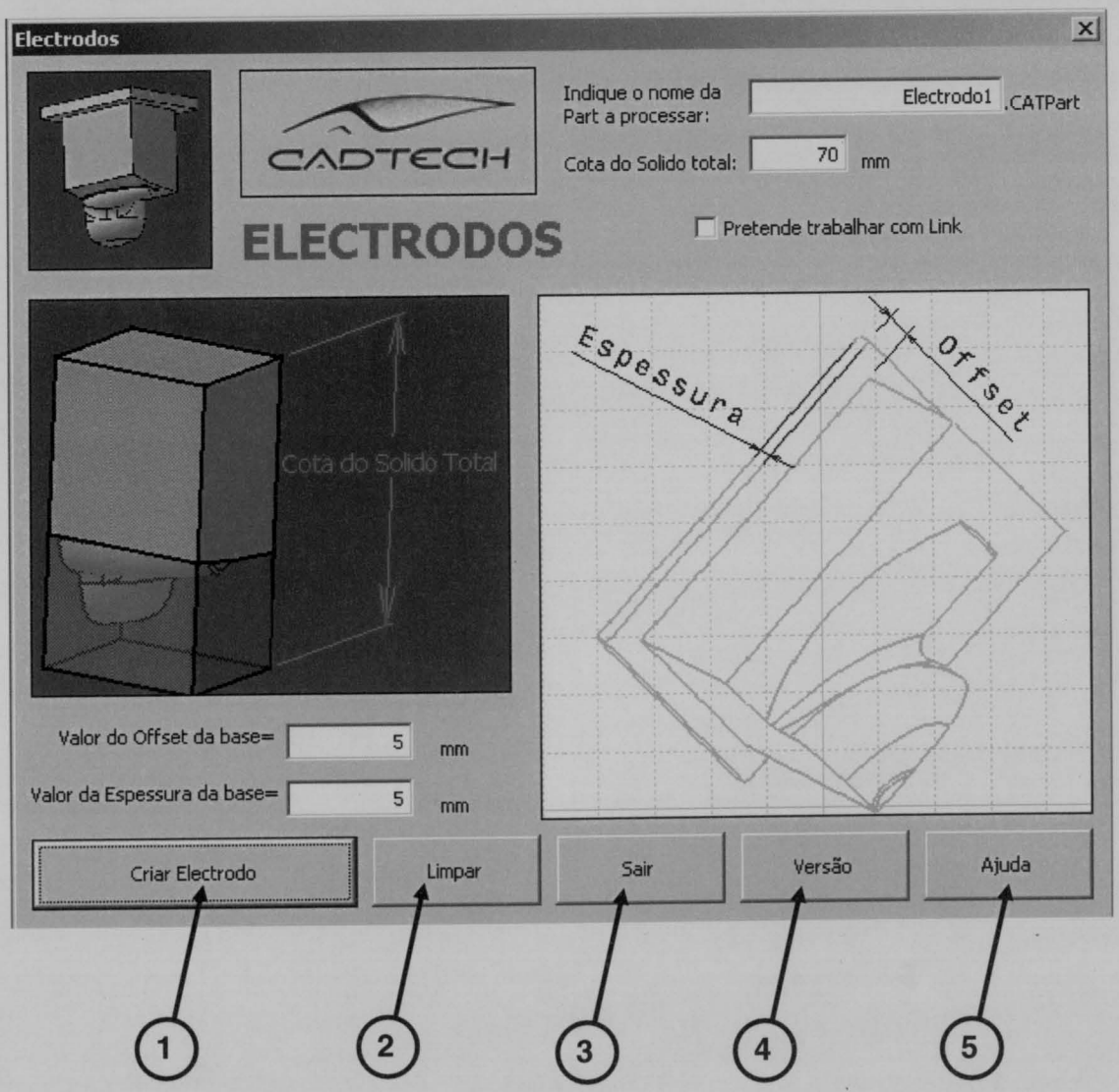


Fig. 54 – Estrutura da árvore do Produto para executar os Eléctrodos

Depois de conceber a Interface, vamos dar início à concepção do programa. Vamos começar por definir primeiro o sólido e o sentido de propagação do *Pad*. Como sabemos, o sentido do *Pad* será um factor importante, visto que, podemos estar a trabalhar com o Macho ou com a Cavidade e sendo assim, como o sistema de eixos do Molde é sempre o mesmo, o sentido de propagação do *Pad* para o Macho e para a Cavidade são diferentes. Para determinar este sentido, vamos ter que fazer com que o utilizador escolha entre o Macho ou a Cavidade. Para este efeito, vamos apresentar um *Form* que contenha esta opção (figura 55).

Fig. 55 – *Form* inicial para escolha do lado da Cavidade ou do Macho

Depois da escolha do lado do Macho / Cavidade vamos ter de implementar uma instrução de selecção, que permite ao programa executar uma ou outra acção, consoante o contexto do programa. Para isto, vamos utilizar a estrutura da instrução ***If*** que permite executar, ou não, um conjunto de instruções conforme a condição seja verdadeira ou falsa.

Ex:

```
If <condição> then
    <instruções>
Else
    <instruções>
End If
```

No caso deste quadro inicial, vamos utilizar *Option Buttons* (Botões de Opção) e vamos ter de impor a condição de estar, obrigatoriamente, activada uma opção ou outra. Para isso vamos utilizar a instrução de outra forma, colocando-a explicitamente a chamar o código do Macho ou o código da Cavidade (figura 56). Esta parte do código será dedicada ao botão ***OK***.

```
Private Sub CommandButton1_Click()

    If indexMacho.Value Then

        F_Electrodo_Macho.Show

    ElseIf indexCavidade.Value Then

        F_Electrodo_Cavidade.Show

    End If

End Sub
```

Fig. 56 – Codificação do botão *OK*

Após a escolha do utilizador, vamos proceder à definição do código em separado quer para o lado da Cavidade, quer para o lado do Macho.

A primeira operação a ser programada será o *Pad*. O método para a obtenção desta operação faz parte do Objecto ***Shape Factory***, que se encontra inserido nos Objectos de Automação ***Part Document*** (figura 57).

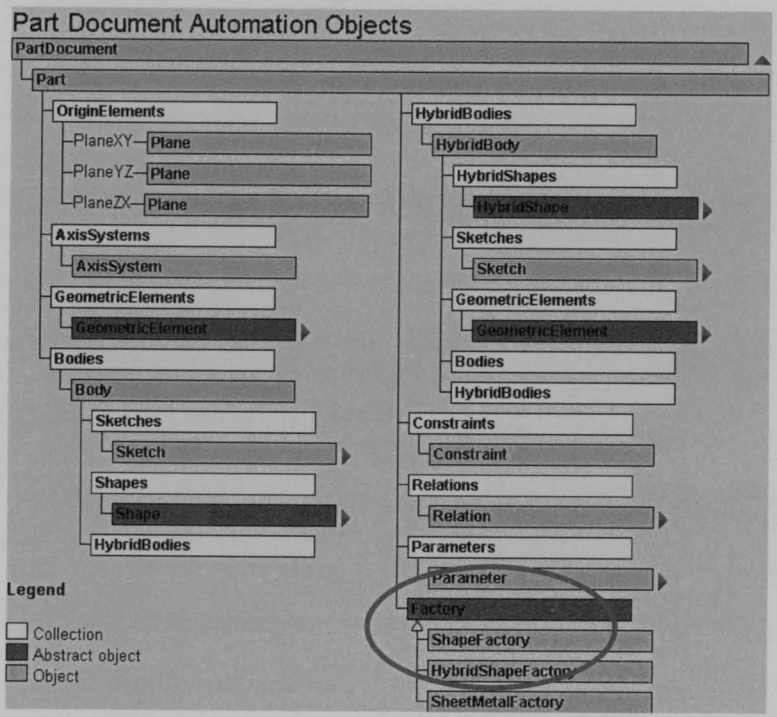


Fig. 57 – Organização dos Objectos *Part Document*

Aqui, a ideia que interessa reter é que a organização dos Objectos segue uma estrutura parecida com a da árvore, estrutura esta que temos vindo a dar ênfase ao longo deste texto. Para aceder ao método para executar um *Pad* temos de aceder a colecção de objectos *Part Document*, depois temos de aceder a colecção de Objectos *Part* e de seguida temos de aceder a colecção de objectos *ShapeFactory* onde vamos ter a opção ***AddNewPad*** (figura 58).

```
Dim documents1 As Documents                                              (1)
Set documents1 = CATIA.Documents

Dim partDocument1 As PartDocument                                        (2)
Set partDocument1 = documents1.Item(nome_electrodo & ".CATPart")

Dim part1 As Part                                                        (3)
Set part1 = partDocument1.Part

Dim shapeFactory1 As ShapeFactory                                        (4)
Set shapeFactory1 = part1.ShapeFactory

Dim pad1 As Pad                                                          (5)
Set pad1 = shapeFactory1.AddNewPad(sketch1, profundidade)
```

Fig. 58 – Estrutura de acesso aos Objectos

Legenda figura 58:

1. *Acesso aos documentos do CATIA*
2. *Acesso aos Objectos "Part Document" para carregar a "Part" que indicamos no início*
3. *Acesso aos Objectos da "Part"*
4. *Acesso aos Objectos da "ShapeFactory"*
5. *Acesso ao método "AddNewPad"*

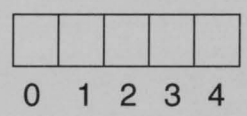
Para o Método ***AddNewPad*** é necessário que se declare a função contendo dois argumentos: o *Sketch* e a Altura do *Pad*, da seguinte forma:

⇨ **AddNewPad** (*iSketch* As Sketch, *iHeight* As Double) As Pad

Esta filosofia será seguida para o resto das operações, entre as quais o ***Split***. Para executar o *Split* vamos ter de indicar qual é a superfície que pretendemos (superfície da Cavidade ou do Macho). Para isto, vamos ter de dar ao utilizador a oportunidade de ser ele a seleccionar qual a superfície que pretende, visto que, a colocação desta superfície na estrutura da árvore pode não ser sempre a mesma, para além de que o próprio nome da superfície poderá, também, não ser sempre o mesmo. Ainda temos que ter atenção ao tipo de superfície, visto que, se estivermos a trabalhar sem *Link* o tipo de superfície será diferente ao tipo de superfície com *Link*. Para poder estabelecer uma certa flexibilidade ao nosso programa, podemos escrever o código de forma a permitir uma selecção interactiva.

Para dar origem a uma selecção interactiva, vamos utilizar o método ***SelectElement***. Este método exige que o argumento que indica qual o tipo de objecto a ser seleccionado, seja um ***Array***. Um *Array* não é mais do que um Vector onde podemos armazenar vários objectos e cuja dimensão é definida pelo número de posições atribuídas no seu dimensionamento. Por exemplo:

```
Dim InputObjectType(4)
```



Na sintaxe do método temos que indicar qual o tipo de superfície (neste caso) que pretendemos seleccionar (figura 59). Ainda podemos adicionar uma mensagem de ajuda que aparecerá na barra inferior do ambiente de trabalho do CATIA.

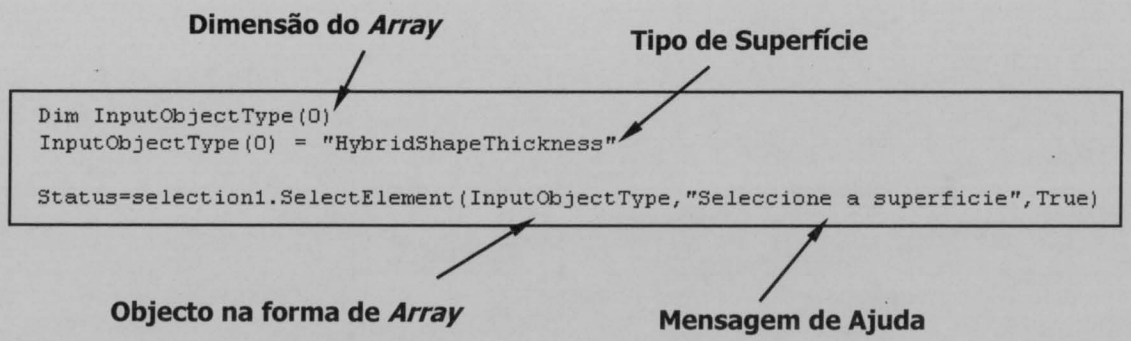


Fig. 59 – Estrutura do método *SelectElement*

Depois da selecção interactiva do elemento para o *Split*, não podemos executar directamente a operação porque estamos a trabalhar em *Parts* diferentes. A *Part* que vai dar origem ao Eléctrodo não é a mesma que contém a superfície de partição.

O problema é que quando fazemos o *Split*, o CATIA automaticamente duplica a superfície e a coloca dentro da *Part* em que estamos a trabalhar, surgindo um erro crítico. A única forma encontrada de ultrapassar este problema, foi conceber uma parte do código que fosse capaz de executar uma cópia idêntica mas separada da original e colocá-la dentro da *Part* que estamos a trabalhar, ou seja, um **"Copy / Paste"**.

Para executar o "Copy / Paste", vamos ter de seguir a mesma filosofia e após seleccionar a superfície, vamos ter de seleccionar a *Part* de destino. A operação de "Copy / Paste" no CATIA V5 é executada à semelhança da mesma operação executada, por exemplo, no Excel. Vamos imaginar que temos um número numa célula e pretendemos copiar esse número para outra célula; o conjunto de operações que executamos para este efeito é o seguinte:

1. *Seleccionar a célula a ser copiada*
2. *Copiar o seu conteúdo para a memória (Copy)*
3. *Seleccionar a célula de destino*
4. *Colar a cópia na célula de destino (Paste)*

Quando programamos esta operação no CATIA a filosofia terá que ser a mesma:

1. *Seleccionar a superfície a ser copiada*
2. *Copiar o objecto para a memória*
3. *Seleccionar a "Part" de destino*
4. *Colar a cópia da superfície na "Part" de destino.*

Da superfície seleccionada podemos extrair o seu nome e assim transformar o objecto que foi seleccionado e copiado, num objecto processável para o método que executa o *Split*. Este método faz parte também do *ShapeFactory* e dá pelo nome de ***AddNewSplit***. Da mesma forma que declaramos o *Pad* podemos declarar o *Split*, apenas mudando os argumentos que neste caso vão ser: o elemento para o *Split* (Superfície) e o lado com que queremos ficar, da seguinte forma:

⇨ **AddNewSplit** (*iSplittingElement* As Reference, *iSplitSide* As CatSplitSide) As Split

É aqui que vamos poder assinalar o lado do sólido com que pretendemos ficar, sendo que, os lados serão diferentes quer para a Cavidade, quer para o Macho.

Existe ainda um problema que não foi resolvido: Como executar a base para o Eléctrodo. Como foi explicado anteriormente, o ideal seria executar um *Offset* do *Sketch* inicial e executar um *Pad* com sentido contrário; no entanto, esta solução revelou-se muito complexa e devido à falta de tempo, esta operação foi executada de outra forma. A solução encontrada parte da colocação de um novo *Body*, onde vamos executar uma projecção do *Sketch* inicial que fará parte deste novo *Body* que corresponderá ao corpo que vai conter a Base. Para executar a projecção vamos utilizar o Método ***CreateProjections*** que faz parte do Objecto *Factory2D*. A sintaxe será a seguinte:

⇨ **CreateProjections** (*iGeometry* As Reference) As GeometricElements

O argumento utilizado é a referência aos elementos resultantes, não do *Sketch* inicial, mas sim da Selecção da Face onde vai estar a base que, no fundo, contêm as arestas que são coincidentes com o *Sketch* inicial. Depois de fechar a edição do *Sketch*, vamos fazer um *Pad* com este *Sketch* resultante no nosso novo *Body*. A seguir a esta operação vamos executar o Método ***AddNewThickness*** que faz parte do Objecto *HybridShapeFactory*. Este método, e à semelhança do comando *Thickness* da *Workbench Part Design* que dá espessura a uma face de um sólido, vai fazer com que a cada uma das faces do *Pad* se dê uma espessura equivalente à distância do *Offset* que inicialmente pretendíamos. A sintaxe é a seguinte:

⇨ **AddNewThickness** (*iSupport* As Reference, *iThickness* As Double, *iThicknessInside* As Double, *iOrientation* As Long) As HybridShapeThickness

Os seus argumentos englobam o suporte para a espessura que, neste caso, é a face, o valor da espessura, o valor da espessura interior (se pretendida) e o sentido da espessura.

Por último falta falar do desenho 2D. Esta aplicação também vai executar um desenho 2D de cada Eléctrodo que, neste caso, utiliza uma folha A3 predefinida da TJ Moldes e onde são inseridas três vistas (Frente, Topo e Esquerda) e uma vista Isométrica (figura 60). Para isto vamos criar um novo módulo onde vamos colocar o código necessário para executar um desenho 2D rápido e o qual vamos chamar a partir do programa principal. Este módulo será aproveitado, também, para fazer o desenho 2D da aplicação seguinte (Postiços).

Resumindo, o utilizador para criar os Eléctrodos, apenas vai ter que criar um novo Produto onde vai inserir a *Molded Part* e irá introduzindo, à medida das necessidades, uma nova *Part* para cada Eléctrodo. Dentro de cada uma destas novas *Parts* o utilizador apenas vai ter que desenhar o *Sketch* para a forma da secção do Eléctrodo pretendido. A partir daqui apenas terá que correr a aplicação, tendo o cuidado de seleccionar correctamente as superfícies respectivas.

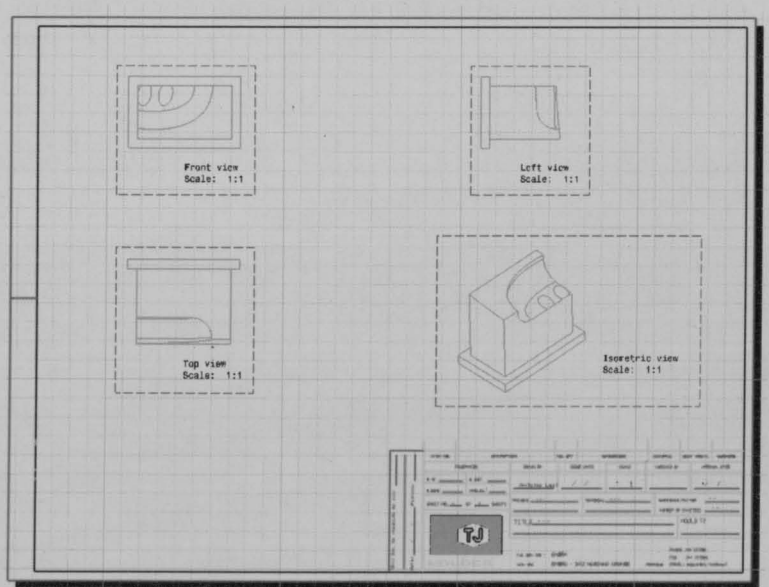


Fig. 60 – Desenho 2D gerado pela Aplicação Eléctrodos

8.4 Breve Referência à Introdução do GAP no Eléctrodo

O Processo de Electroerosão (*EDM* - Electrical Discharge Machining), é um processo de corte no qual a remoção de material é realizada através de uma série de descargas eléctricas sucessivas de curta duração, que se estabelecem entre um Eléctrodo e a peça (sem que haja contacto entre eles), encontrando-se o conjunto imerso num líquido dieléctrico. Este processo é especialmente adaptado para a erosão de aços de elevada dureza para moldes, insertos, machos e cavidades moldantes, para os quais a maquinagem pelos processos convencionais de corte é tecnicamente difícil e lenta. Permite maquinar materiais endurecidos, evitando a necessidade de realizar tratamentos térmicos após maquinagem, o que normalmente se traduz em perda de exactidão dimensional. Neste processo o custo de fabrico dos Eléctrodos representa uma fatia substancial do custo total da maquinagem, sendo reportados valores de 20 a 50% desse custo total, especialmente quando os Eléctrodos apresentam formas de alguma complexidade.

Todos os Eléctrodos necessitam que exista uma distância na zona ao longo da qual se dá o processo de Electroerosão. Esta distância é denominada por *Gap*. Após uma visita à TJ Moldes, pudemos verificar que a introdução desta distância é realizada na própria máquina que executa a maquinagem dos Eléctrodos, sendo que, o modelo 3D fica com as cotas nominais e é no software CAM que se atribui uma cota para o acabamento, que neste caso tomará o valor negativo para poder aplicar uniformemente a toda a superfície esta distância / afastamento (figura 61).

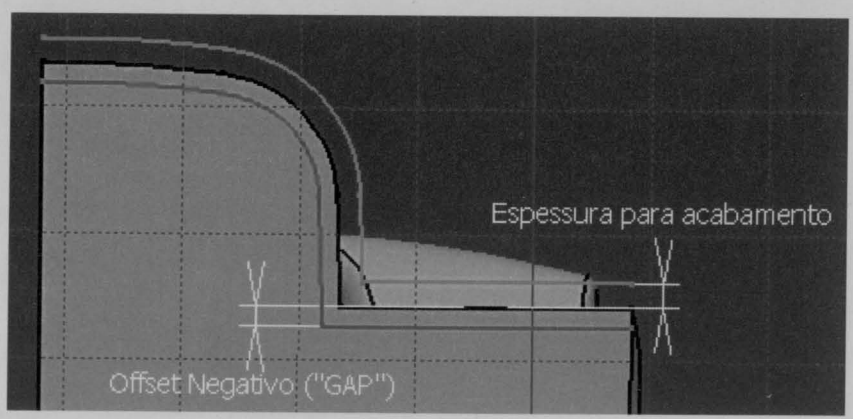
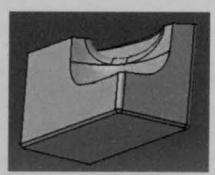


Fig. 61 – Dimensão dada na maquinagem do Eléctrodo com valor negativo, da qual resulta o *GAP*

Concepção de uma Ferramenta para a Execução de Postiços Para um Molde



9

9. Concepção de uma Ferramenta para a Execução de Postiços para um Molde

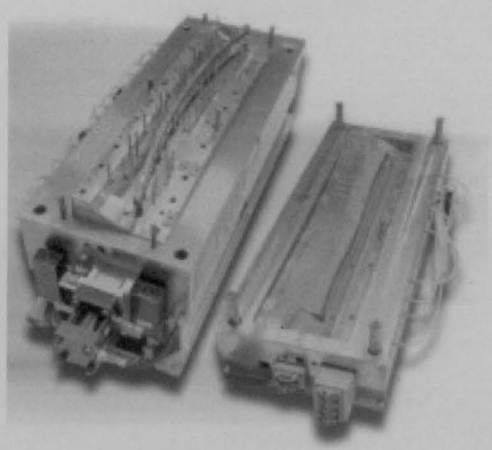
9.1 Visita à SOMEMA

Outra das empresas que foi visitada, foi a **SOMEMA**. Fundada em 1958 e localizada na Marinha Grande, a empresa SOMEMA - Sociedade Metalúrgica Marinhense, Lda, orientou desde o início a sua actividade para a concepção e produção de Moldes para a indústria transformadora de matérias plásticas. O consistente crescimento e desenvolvimento da empresa, levou a SOMEMA a assumir-se também como parceiro estratégico dos seus clientes, desempenhando hoje um papel importante nas fases de inovação, concepção, desenvolvimento de produto e realização de protótipos, nomeadamente para os seus clientes ligados à Indústria Automóvel. A SOMEMA é uma empresa constituída por uma equipa de 75 pessoas que faz da satisfação plena dos seus clientes o objectivo diário da sua actividade. A SOMEMA é uma das empresas certificadas pelo Sistema Nacional de Gestão da Qualidade, de acordo com a norma ISO 9001. Para melhor responder às exigências dos clientes, espalhados um pouco por todo o mundo e em particular na Europa e na América, continentes que absorvem 95 % da sua produção, a SOMEMA utiliza os mais exigentes materiais e as mais avançadas tecnologias existentes no mercado.

A SOMEMA é especialista no fabrico de Moldes para a Injecção de plástico, de alta precisão, e para máquinas com uma força de fecho superior a 300 toneladas.

Tipos de Moldes produzidos:

- ⇨ Injecção convencional
- ⇨ Bi-Material com sistema de levantamento hidráulico
- ⇨ Bi-Material rotativo
- ⇨ Bi-Color
- ⇨ Injecção sequencial com válvula de entrada para moldação com têxteis
- ⇨ Injecção combinada: Têxtil + gás
- ⇨ Injecção combinada: Têxtil + Bi-Material



9.2 Necessidades da Empresa

É importante referir que, a aplicação anterior não se destina única e exclusivamente à empresa em causa, mas poderá ser adaptada a um novo cliente tendo em conta a metodologia da execução do Eléctrodo. Poderá talvez, ser utilizada a mesma aplicação para um novo cliente, podendo haver a necessidade de fazer alguns ajustes no programa de modo a personalizar as operações à metodologia de trabalhar da empresa.

Para além da aplicação referida no capítulo anterior, na visita à SOMEMA surgiu a ideia de conceber uma aplicação para fazer Postiços na modelação do Molde. A metodologia de modelação dos Postiços é semelhante à dos Eléctrodos, no entanto, como vamos poder ver, existem algumas variantes que tornam esta aplicação um pouco mais complexa do que a dos Eléctrodos. Existe de facto, uma série de tarefas que permitem a execução de um Postiço num Molde. Estas tarefas podem ser contidas numa aplicação na qual se introduza alguns parâmetros e se obtenha o Postiço com a forma final, contido dentro do Molde e com a respectiva caixa.

Outra ideia que surgiu, foi a de fazer outra aplicação que conseguisse tirar as dimensões máximas de, por exemplo, um Postiço. Quando falamos de dimensões máximas, estamos a falar de atravancamento do Postiço no Molde. A complexidade deste algoritmo e a limitação de tempo para realizar uma ferramenta que no fundo executa uma operação rápida e fácil, fez com que se abandonasse logo à partida esta ideia.

Um dos grandes problemas que surge na modelação 3D em quase todos os softwares semelhantes ao CATIA é o desenho 2D. Ainda se encontra muito enraizado o hábito dos moldistas de, após a conclusão da modelação 3D, não perderem tempo com o 2D do próprio programa 3D e passarem todo o desenho para AutoCAD. Existe ainda a ideia de que o software falha no que diz respeito ao desenho 2D; no caso do CATIA, quando se faz um desenho 2D, a partir duma peça 3D, surge um problema de sobreposição de linhas, que de resto é comum a todos os softwares. Este problema diz respeito à interactividade do 3D com o 2D, visto que em CATIA V5 o desenho está ligado ao Modelo 3D. Como é evidente, em certas situações como a da figura 62, o software não entende a sobreposição de linhas como algo de errado, mas sim como algo de natural que resulta de elementos que realmente existem, mas a sua projecção é em todo desnecessária.

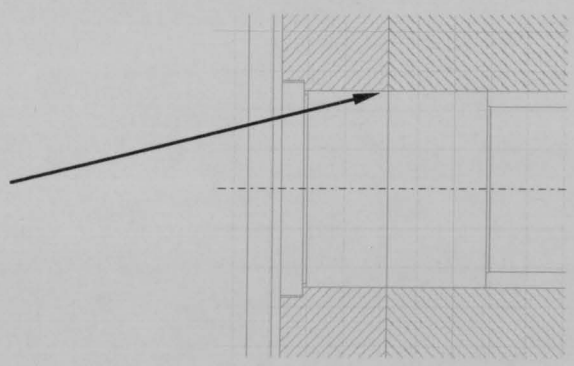


Fig. 62 – Sobreposição de linhas quando se faz um corte

Esta situação, motivou o surgimento da ideia de se fazer uma aplicação que conseguisse "limpar" o desenho 2D, destes elementos que estão a sobrecarregar o desenho sem necessidade. Esta ideia não teve seguimento para já, visto que se chegou à conclusão que não se pode criar uma *Macro* para corrigir "erros" do software de base.

9.3 Concepção e Desenvolvimento da Aplicação

À semelhança da aplicação anterior, é necessário conhecer todas as etapas que levam à Modelação final de um Postiço. Para começar, é necessário saber como é um Postiço num Molde.

Um Postiço do tipo que vamos modelar, não é mais do que uma peça que faz parte da placa moldante (quer seja da Cavidade ou do Macho) que é realizado em separado do resto da placa e encaixa na perfeição no sítio de onde foi extraído. Normalmente situa-se numa zona frágil ou de difícil maquinagem. Esta peça pode ser, por exemplo, feita de um material diferente, por razões de melhor escoamento de calor nessa zona, beneficiando das propriedades diferentes do material.

Um Postiço não é apenas um paralelepípedo, pois não existem arestas vivas tanto na cavidade onde vai ser encaixado, como na própria peça. Um Postiço deste tipo, pode ter um arredondado nas arestas laterais e um chanfro nas arestas da face inferior, sendo que na cavidade onde vai encaixar deverá conservar um arredondado idêntico nas arestas laterais e um arredondado nas arestas da face inferior (figura 63).

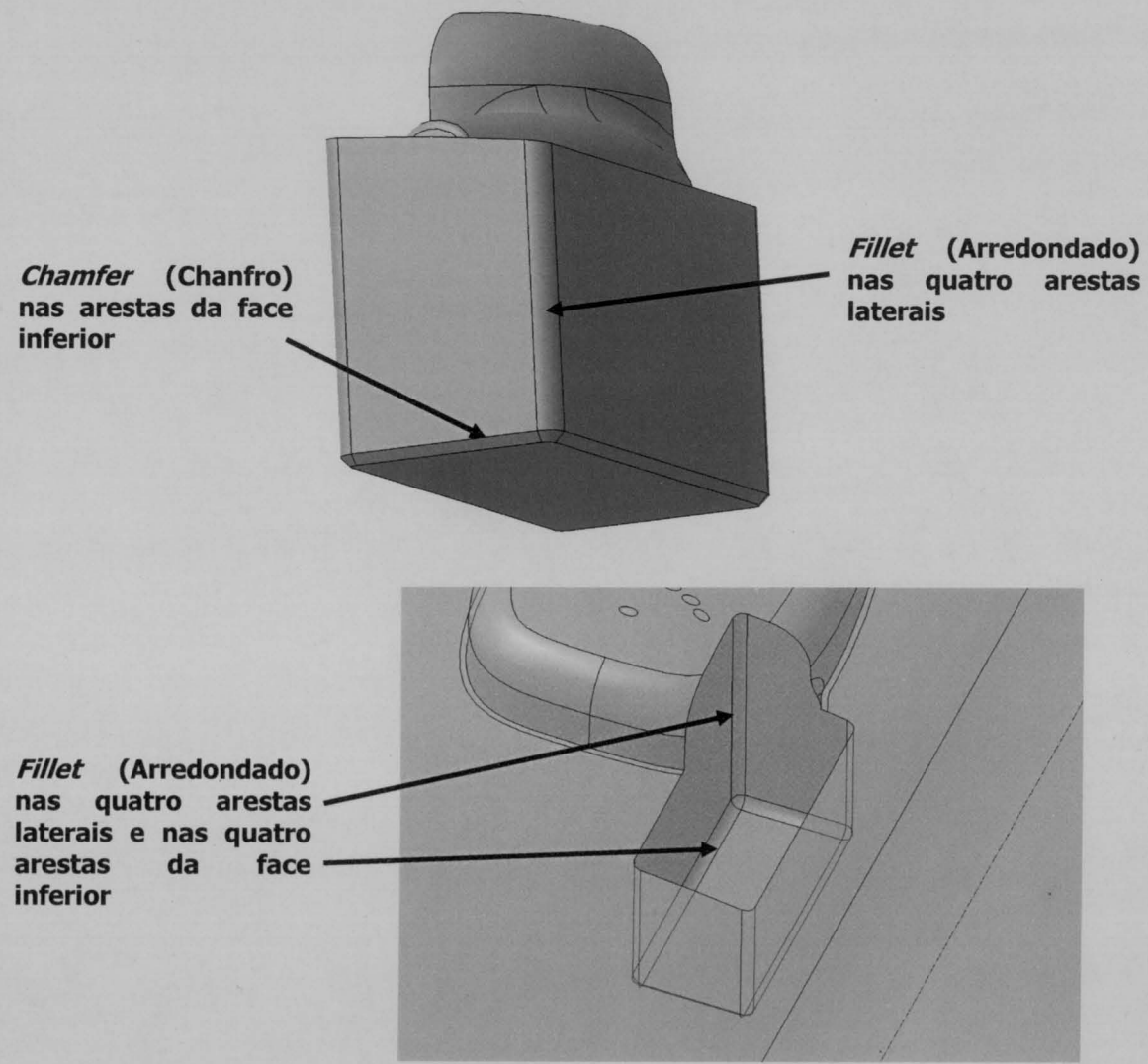


Fig. 63 – Postiço para uma zona moldante específica

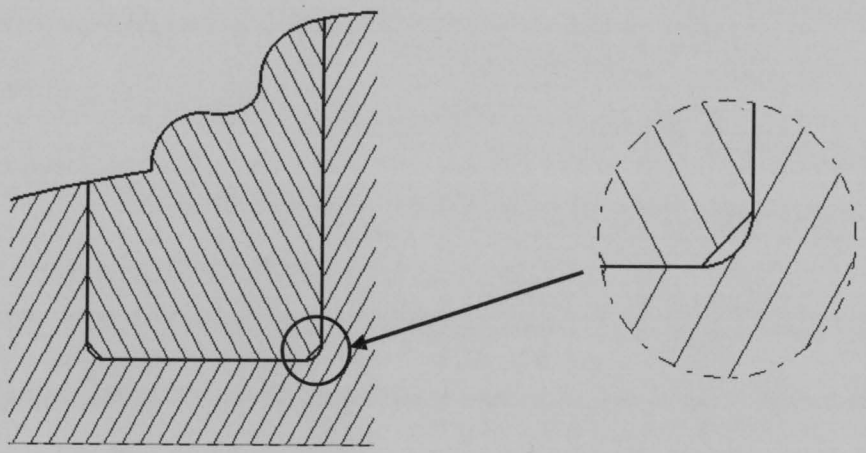


Fig. 64 – Corte de um Postiço inserido no macho, para mostrar a zona do canto

De seguida vamos descrever a série de operações que levam à obtenção da modelação de um Postiço. Para começar, vamos ter de verificar a situação que referimos no capítulo anterior, relativamente ao sistema de eixos. Tal como no caso anterior, temos que verificar se existe a necessidade da colocação de um plano auxiliar num ponto mais alto do que a superfície que vai dar origem à separação do sólido (Split).

O passo seguinte é o de colocar uma nova *Part* no nosso Produto. Neste passo, é necessário ter atenção que, ao contrário do caso anterior, vamos trabalhar directamente com o Molde aberto, visto que, ao contrário do Eléctrodo, esta *Part* vai ser parte integrante do Molde e deverá ser concebido como se fosse um componente que exista no MTD (figura 65). Depois de inserir a *Part*, vamos definir a forma da secção do nosso Postiço começando por desenhar um *Sketch*.

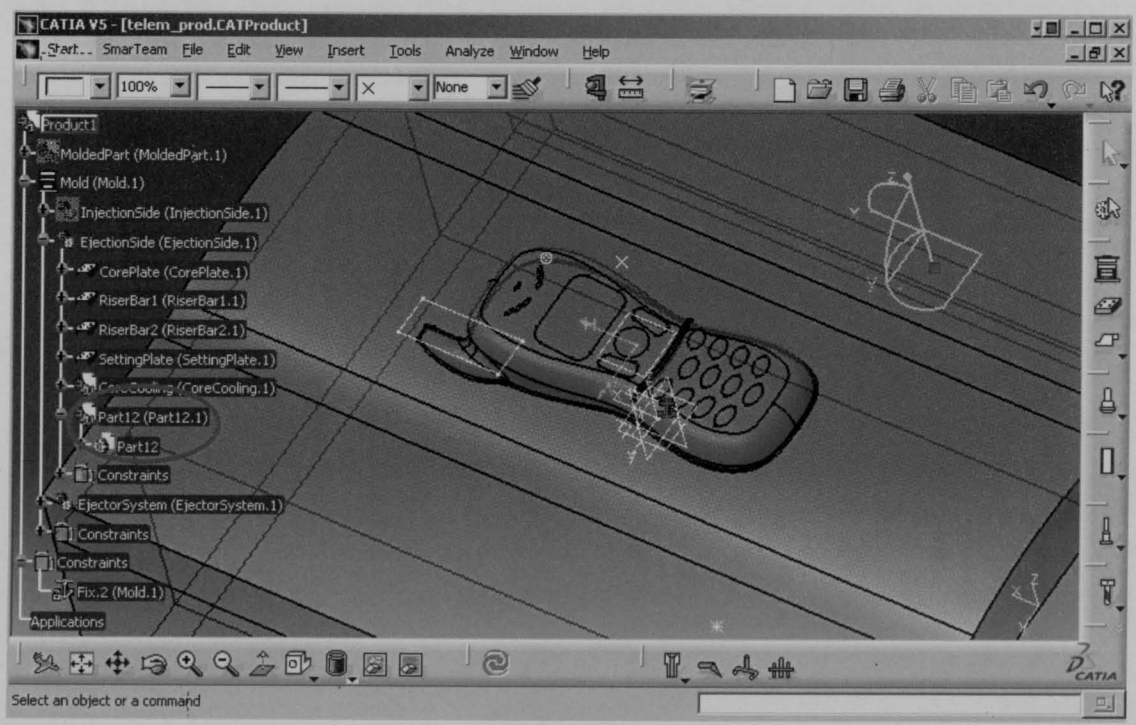


Fig. 65 – Colocação de uma nova *Part* directamente na estrutura do Molde

O passo seguinte é criar um *Pad* com o comprimento necessário para que a superfície de partição corte na totalidade o sólido, e o pretendido para conservar uma distância até ao fundo da placa para permitir fixar (geralmente por parafusos) o Postiço. Assim, Este *Pad* terá o comprimento, definido pelo utilizador, de modo a garantir uma distância ao fundo da placa que servirá e de base de assentamento (figura 66).

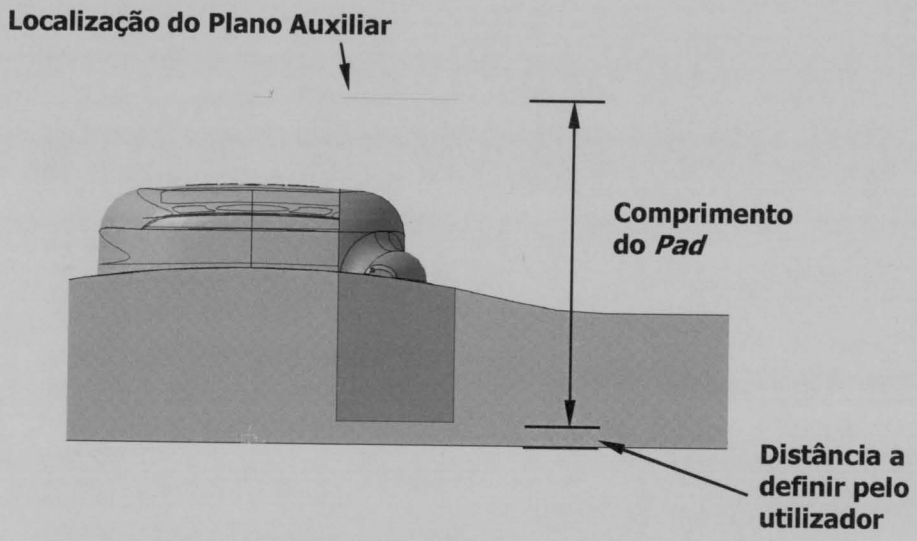


Fig. 66 – Distâncias definidas pelo utilizador

Agora estamos em condições de poder efectuar o ***Split***  . Esta operação será executada aproveitando as superfícies de partição do Molde, que se encontram dentro da *Molded Part*. No caso deste Postiço, vamos utilizar a superfície *Core Side*. O sentido será definido dependendo de estar a trabalhar com o Macho ou com a Cavidade, tendo em conta que a parte com que queremos ficar é a parte que está dentro da placa (figura 67).

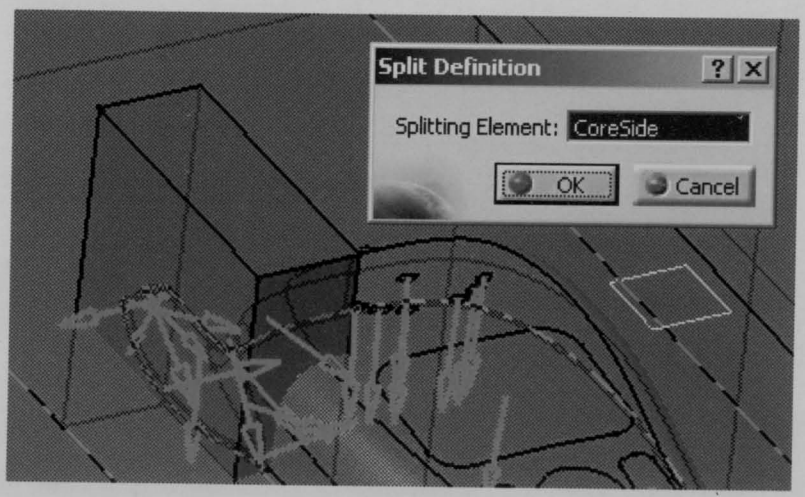


Fig. 67 – Definição do *Split*

Depois de fazer o *Split*, vamos ficar com um sólido que será o nosso Postiço preliminar (figura 68). De seguida vamos modelar este sólido de modo a obter o Postiço final. Para isto, vamos começar por dar saída a algumas faces do Postiço. Esta operação é necessária para proporcionar um melhor ajustamento (sendo que o ajustamento terá que ser o mais perfeito possível, para diminuir ao máximo as rebarbas deixadas na peça a injectar), e para delimitar a superfície de deslizamento entre o material do Postiço e o Aço da placa, facilitando assim a sua montagem / desmontagem.

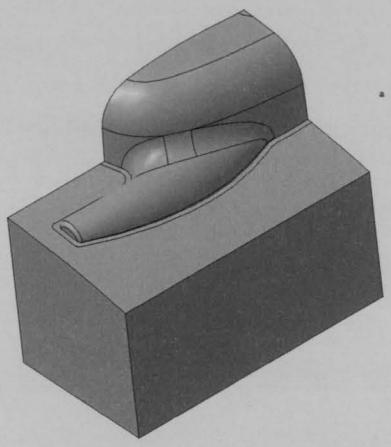


Fig. 68 – Postiço preliminar

Esta operação será executada com o comando ***Draft Angle***  , que faz parte da *Workbench Part Design.* O ângulo de *Draft* será definido pelo utilizador, bem como o número de faces a que se dá essa saída. A direcção será sempre a da figura 69. Neste caso, só vamos dar saída a duas faces mas poderá ser necessário seleccionar mais faces.

- Faces com saída

- Face neutra

Fig. 69 – Faces, Direcção e ângulo de *Draft*

Agora, podemos atribuir um arredondado às quatro arestas dos cantos do sólido, como representado na figura 70. Esta operação será executada com o comando ***Edge Fillet*** , que também faz parte da *Workbench Part Design*.

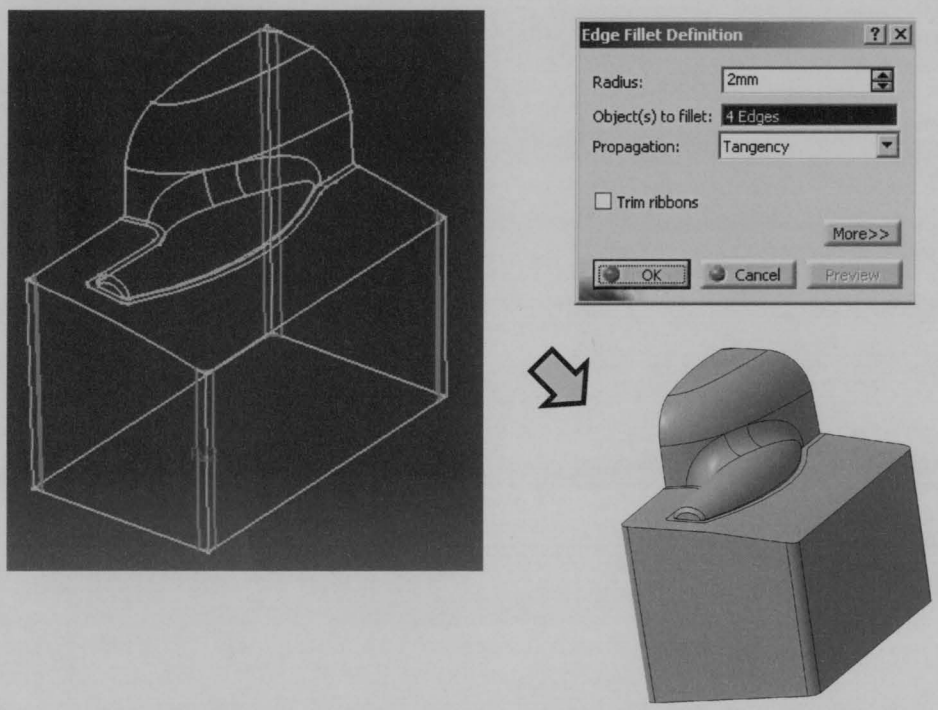


Fig. 70 – Definição do *Edge Fillet*

O próximo passo será utilizar o comando ***Chamfer*** para executar um chanfro ao longo das arestas da face inferior do nosso sólido. Este chanfro poderá ser propagado automaticamente a todas as arestas da face inferior, dado que após a execução do *Fillet*, passou a existir continuidade em tangência ao longo dessas arestas (figura 71).

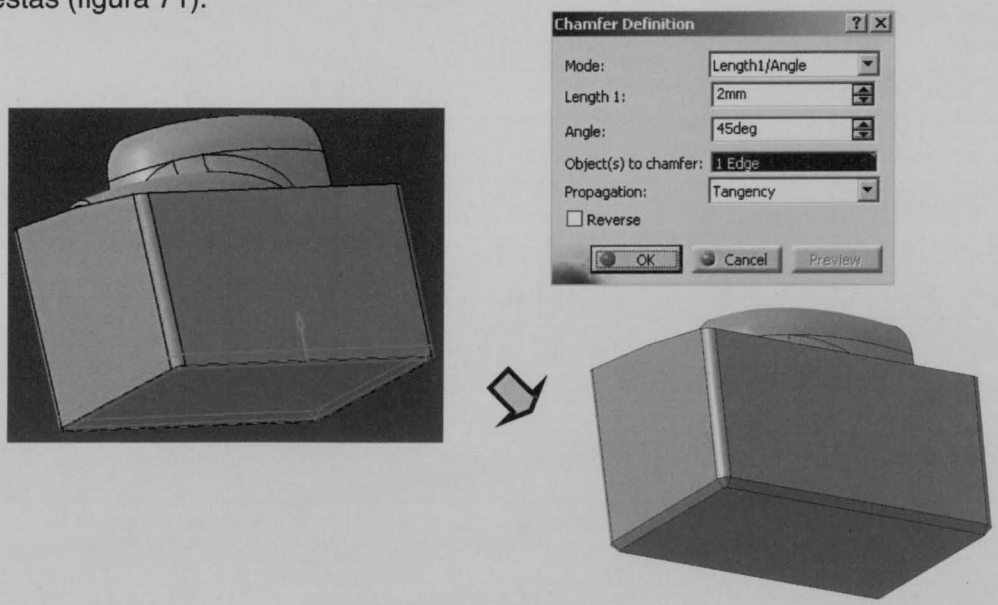


Fig. 71 – Definição do *Chamfer*

Nesta fase é necessário realçar um aspecto importante. Se o raio do *Fillet* for inferior à cota do Chanfro, existirá um problema de tangência visto que quando se dá a propagação do Chanfro ao longo das arestas, na zona dos cantos (zona na qual após o *Fillet*, passamos a ter mais quatro arcos) acontecerá um erro visto que as superfícies resultantes se cruzam (figura 72). Logo o valor do Chanfro, não deverá ser superior ao valor do raio do *Fillet*.

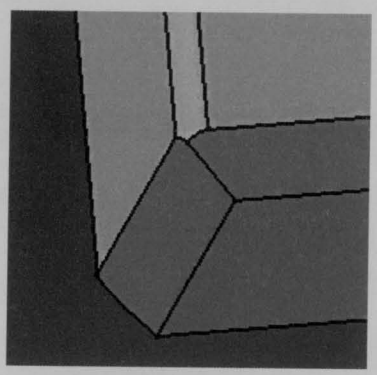
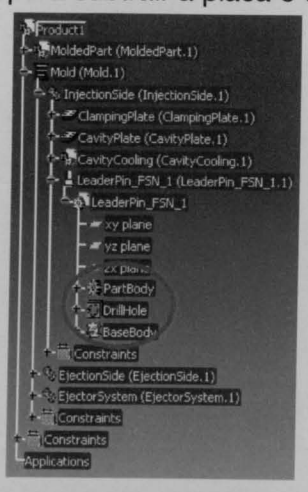


Fig. 72 – Chanfro errado

Estamos agora em condições de abordar o problema de executar a caixa para o Postiço na Placa (neste caso, no Macho). Para isto vamos ter de estudar a forma mais correcta para o fazer. Existem duas formas de executar esta operação. A primeira é executar operações Boleanas, como por exemplo um *Remove*. Para executar um *Remove*, vamos necessitar de um corpo idêntico ao que temos vindo a modelar. Este corpo terá de se encontrar dentro da nova *Part* que criamos e estará sobre a forma de um *Body*. Em termos da metodologia do MTD, esta operação pode ser executada automaticamente, e por esta razão a opção tomada não foi esta.

A segunda opção, será seguir à risca a metodologia do MTD. Sendo assim, vamos tratar o Postiço criado como sendo um "componente". Dentro do MTD, existe um comando que se chama ***Drill Component***; este comando consegue efectuar a operação de remoção de material automaticamente. Para este efeito, decidiu-se seguir a solução adoptada para as peças *Standard*, assim, foi necessário estudar a estrutura da árvore de um componente existente num catálogo interno do MTD, sendo que o componente utilizado foi uma Guia (figura 73). A estrutura da árvore deste componente revela que existe um *Part Body* e um *Open Body* (que no caso da Guia se vai chamar *BaseBody*, e vai conter apenas a referência do ponto de inserção), que são elementos que estão presentes também, na árvore do nosso Postiço. Na figura 73, é possível reparar que existe outro corpo presente. Este corpo chama-se ***DrillHole***, e é o corpo negativo que foi mencionado no capítulo 3. Este corpo é idêntico ao *PartBody* e serve para subtrair à placa o espaço necessário para a Guia.



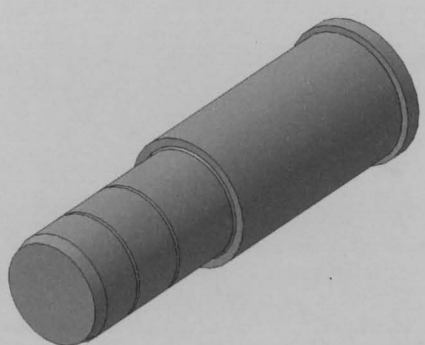


fig. 73 – Estrutura de árvore de uma Guia

Então, vamos ter de criar este novo corpo na estrutura do nosso Postiço. Para isto, vamos inserir um novo *Body* dentro da *Part* do Postiço. Se aproveitarmos o mesmo *Sketch* a partir do qual fizemos o Postiço, podemos utilizar o comando ***Pocket*** com a mesma distância que demos ao *Pad* inicial. Sendo assim, vamos poder reparar que o ícone do novo *Body* se alterou, e passou a ser um ícone representativo da operação negativa que estamos a fazer (figura 74).

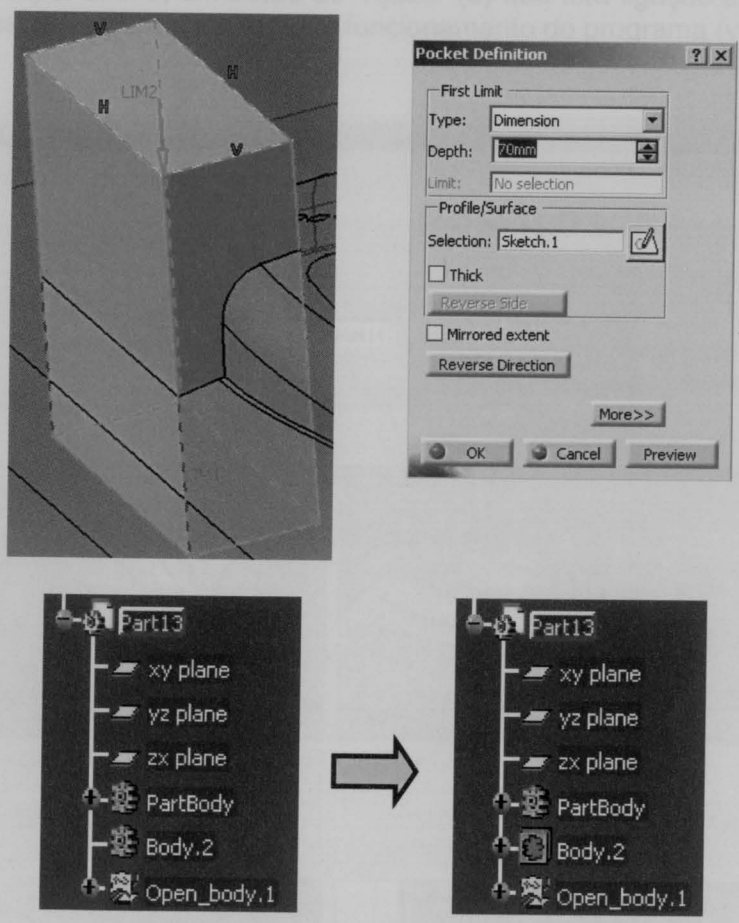


Fig. 74 – Execução do *Pocket*

Podemos reparar que, o nome do novo *Body* não mudou. Para executar o *Drill Component* o nome deste novo corpo vai ter de passar a ser ***DrillHole***, e se assim não for a operação vai falhar. Este é um problema que vamos ter de resolver no desenvolvimento do programa. Ainda, temos de executar o Fillet nas arestas do interior da caixa, ou seja, as arestas que vão formar a caixa no corpo do *DrillHole*.

Após todas estas considerações sobre a sequência de operações, podemos passar ao desenvolvimento da aplicação.

Como foi referido no capítulo anterior, vamos ter de definir a interface com o utilizador. Esta interface será mais extensa do que a anterior, visto que, como realizamos mais operações, vamos ter novos campos para mais alguns parâmetros.

Vamos começar por colocar um botão que dê início à aplicação chamado "Criar Postiço" (**1**). Este botão conterá dentro de si, a maior parte do código. À semelhança da aplicação anterior, vamos colocar mais quatro botões; Um para "Limpar" (**2**) todos os campos ao mesmo tempo, outro para "Sair" (**3**) da aplicação, outro para indicar qual a "Versão" (**4**) e por último, um botão de "Ajuda" (**5**) que terá ligação a um ficheiro de texto com a descrição pormenorizada do funcionamento do programa (ver figura 75).

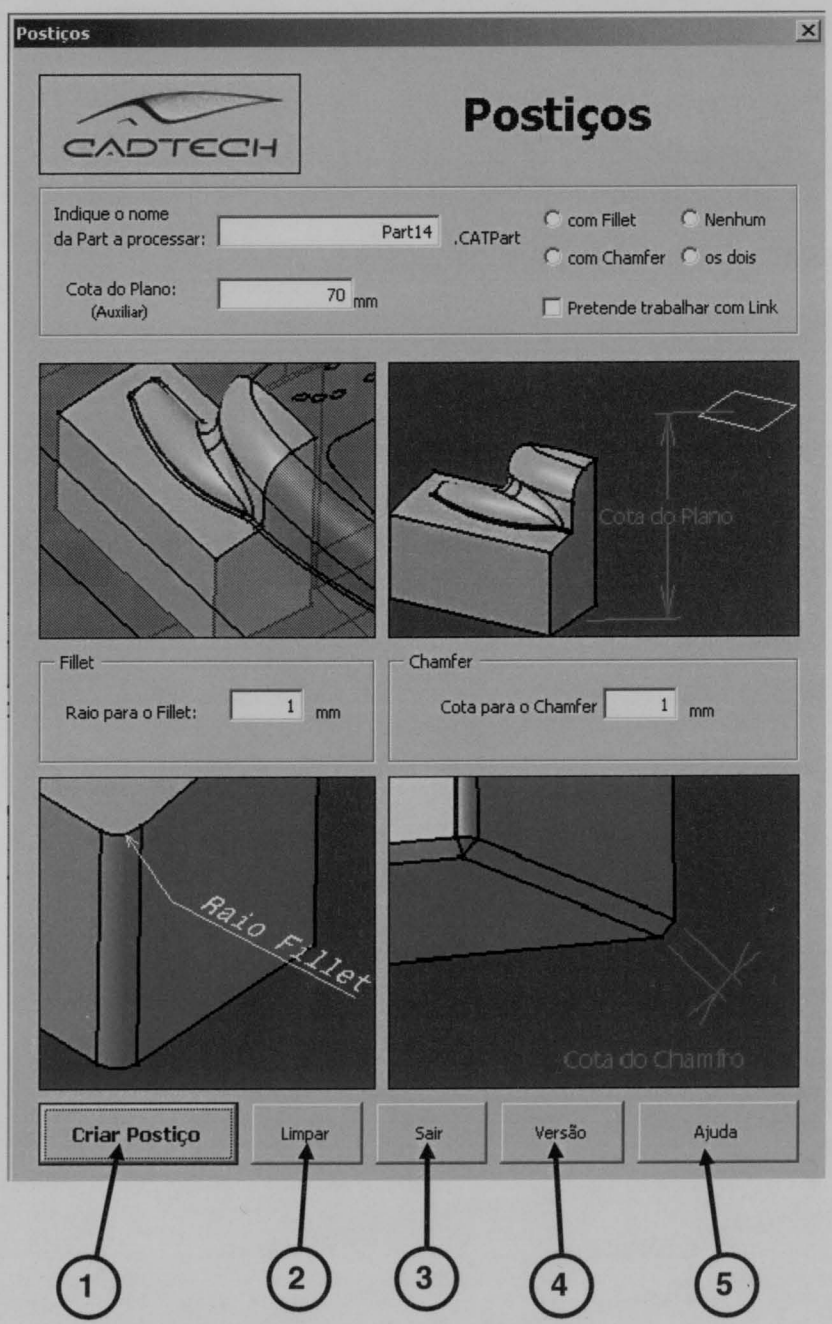


Fig. 75 – Interface da aplicação Postiços

Para além dos botões referidos, vamos ter um campo onde vamos introduzir o nome da *Part* a ser processada, um campo para introduzir a **"Cota do Plano"**, um campo para introduzir a medida do Chanfro e, por último, um campo para introduzir o raio do *Fillet*.

Depois de concebida a interface, podemos passar à concepção do programa. As operações a programar, que são em todo semelhantes às do caso do capítulo anterior, apenas diferem no que diz respeito ao lado do sólido com que pretendemos ficar que, neste caso, será o contrário ao do Eléctrodo. Neste caso, o lado com que queremos ficar é o lado do sólido que está dentro da placa. De resto, as operações são as mesmas.

No caso dos Postiços, vamos ter também, um *Form* inicial onde o utilizador vai poder escolher entre processar o lado do Macho ou o lado da Cavidade (figura 76).

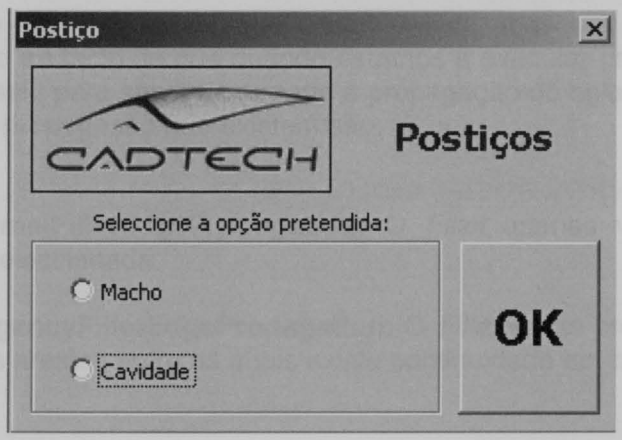


Fig. 76 – *Form* inicial para escolha do lado a trabalhar, da Cavidade ou do Macho

A utilidade deste ecrã inicial já foi explicada anteriormente e tem a mesma finalidade. As operações para executar o *Pad* e o *Split* são iguais, sendo que no *Split* vamos dizer que queremos o lado oposto ao que indicamos nos Eléctrodos. Para indicar qual o lado com que queremos ficar, vamos alterar um dos argumentos do método ***AddNewSplit***, da seguinte forma:

⇨ Declaração para o *Split* do Postiço, lado da Cavidade

```
Dim hybridShapeThickness1 As HybridShapeThickness
Set hybridShapeThickness1 = hybridShapes1.Item("Result of " & nomeobjecto)

Dim split1 As Split
Set split1 = shapeFactory1.AddNewSplit(hybridShapeThickness1, catPositiveSide)
```

⇨ Declaração para o *Split* do Postiço, lado da Macho

```
Dim hybridShapeThickness1 As HybridShapeThickness
Set hybridShapeThickness1 = hybridShapes1.Item("Result of " & nomeobjecto)

Dim split1 As Split
Set split1 = shapeFactory1.AddNewSplit(hybridShapeThickness1, catNegativeSide)
```

As grandes diferenças começam aqui, pois vamos ter de definir as operações para executar os boleados e os chanfros. Vamos começar pelo boleado (Fillet). O método para executar um *Fillet* também faz parte do *ShapeFactory*, e dá pelo nome de ***AddNewEdgeFilletWithConstantRadius***.

Neste método vamos ter de declarar três argumentos, entre os quais a aresta que queremos processar, o tipo de propagação e o valor do Raio, da seguinte forma:

⇨ **AddNewEdgeFilletWithConstantRadius** (*iEdgeToFillet* As Reference, *iPropagMode* As CatFilletEdgePropagation, *iRadius* As Double) As ConstRadEdgeFillet

Neste caso não é a aresta que nos é pedida, mas sim a referência desta aresta. Isto deve-se ao facto de que quando estamos a executar um *Fillet*, o programa só necessita da aresta para servir de suporte à propagação do boleado.

Os tipos de propagação que existem são:

- **CatMinimalFilletEdgePropagation:** O *Fillet* apenas vai-se propagar na aresta seleccionada.
- **CatTangencyFilletEdgePropagation:** O *Fillet* vai-se propagar ao longo de todas as arestas entre as quais exista continuidade em tangência.

Para que só exista um código para cada vez que se executa um *Fillet*, esta parte do código foi colocada sob a forma de um módulo que fica em separado do resto do código principal e que pode ser reutilizado as vezes que for necessário. Ainda existe o problema da selecção das arestas. O utilizador tem que ser capaz de seleccionar quais são as arestas que pretende com *Fillet*. Para isto, vamos utilizar de novo o método ***SelectElement*** para executar a selecção interactiva de uma aresta. Este método é em todo semelhante ao explicado no capítulo anterior apenas varia o tipo de objecto que estamos a seleccionar, que neste caso terá o nome de ***Edge***.

Ainda é necessário conceber uma forma de seleccionar várias arestas e processar uma a uma sem sair da operação *Fillet*. Para isto, vamos ter de criar uma instrução de repetição. A instrução de repetição utilizada será o ***For ... Next*** . Esta instrução é declarada da seguinte forma:

```
For <contador> = <início> to <fim>
    <instruções>
Next
```

Assim, o utilizador vai seleccionar uma a uma todas as arestas que pretende com *Fillet*, sendo que, lhe será sugerido em cada passo a oportunidade de parar com o *Fillet* e prosseguir com o resto do programa.

A operação que se segue será o Chanfro. Para isto, vamos criar também um módulo à parte para que exista a possibilidade de reutilizar esta parte de código. A operação do Chanfro é em todo semelhante à operação para obter o *Fillet*. O método para executar um Chanfro também faz parte do *ShapeFactory*, e dá-se pelo nome de ***AddNewChamfer***.

Neste método vamos ter de declarar três argumentos, entre os quais a aresta que queremos processar, o tipo de propagação, o tipo de Chanfro, o sentido de orientação, a dimensão do Chanfro e o valor do ângulo, da seguinte forma:

⇨ **AddNewChamfer** (*iObjectToChamfer* As Reference, *iPropagation* As CatChamferPropagation, *iMode* As CatChamferMode, *iOrientation* As CatChamferOrientation, *iLength1* As Double, *iLength2OrAngle* As Double) As Chamfer

Os tipos de propagação que existem são:

- **CatMinimalChamfer:** A propagação far-se-á apenas na aresta seleccionada
- **CatTangencyChamfer:** A propagação estender-se-á às arestas onde houver continuidade em tangência

Os tipos de Chanfros que existem são:

- **CatLengthAngleChamfer:** Chanfro definido por uma distância e um ângulo
- **CatTwoLengthChamfer:** Chanfro definido por duas distâncias

Os tipos de orientação que existem são:

- **CatNoReverseChamfer**
- **CatReverseChamfer**

Por fim, vamos ter de criar o tal corpo "negativo". Esta operação executa-se adicionando um novo *Body* dentro da mesma *Part*. Este novo *Body* terá o nome de *DrillHole*. Este nome é obrigatório, visto que sem ele o *DrillComponent* não o vai identificar. Para isto vamos ter de alterar o nome do novo *Body* acedendo as propriedades do objecto em causa. Depois vamos ter que executar um *Pocket* neste novo *Body* com o mesmo *Sketch* do *Pad*. Para este efeito, vamos utilizar o método ***AddNewPocket*** que também faz parte do *ShapeFactory*. A sintaxe fica desta forma:

⇨ **AddNewPocket** (*iSketch* As Sketch, *iHeight* As Double) As Pocket

Aqui o *Sketch* será o mesmo que indicamos para o *Pad* e a altura será exactamente a mesma que demos ao Sólido inicial. Ao correr o programa, vamos poder constatar que o ícone do novo *Body* mudou de acordo com o que pretendíamos.

Ainda nos falta declarar a operação para dar as saídas (executar o *Draft*). O método para executar esta operação chama-se ***AddNewDraft*** e também o podemos encontrar no *ShapeFactory*. A sua sintaxe é a seguinte:

⇨ **AddNewDraft** (*iFaceToDraft* As Reference, *iNeutral* As Reference, *iNeutralMode* As CatDraftNeutralPropagationMode, *iParting* As Reference, *iDirX* As Double, *iDirY* As Double, *iDirZ* As Double, *iMode* As CatDraftMode, *iAngle* As Double) As Draft

Os argumentos deste método são os seguintes: em primeiro lugar a face à qual se pretende dar a saída, a face neutra, a forma de propagação, uma aresta para indicar o ponto de partida do *Draft* e as três coordenadas que definem um vector que será o sentido de abertura da saída. Devido a falta de tempo não foi possível programar esta operação. No entanto, é necessário ter atenção que esta operação deve ser realizada antes dos *Fillets* e dos Chanfros, visto que no caso dos *Fillets* se esta operação for realizada depois podemos ter o problema de não ter um raio constante ao longo das arestas pretendidas.

O desenho 2D deste componente também é executado após a conclusão de todas as operações, sendo que, também é colocado numa folha predefinida da empresa em causa. As Vistas são as mesmas da aplicação do capítulo anterior (vista de frente, topo, esquerda e isométrica). Para isto vamos aproveitar o Módulo que criamos no capítulo anterior e colocá-lo directamente dentro da aplicação em causa, sendo que, vamos mudar apenas o tipo de folha.

Depois de obter a peça final vamos ainda ter de executar uma operação chamada ***DrillComponent***, onde vamos subtrair o nosso *DrillHole* à placa em causa, ficando com o espaço vazio para a inserção do Postiço na placa.

Como resultado final, vamos criar um novo componente dentro do Molde, exactamente igual a um colocado pelo MTD. Apenas é necessário colocar uma nova *Part* no sítio correcto, desenhar um *Sketch* com a forma pretendida para o Postiço e ter atenção ao sítio onde estamos a trabalhar e onde vamos buscar as superfícies.

9.4 Comparação com o *Inserts* do MTD

No Mold Tooling Design do CATIA V5 existe uma ferramenta parecida com a que desenvolvemos. Esta chama-se ***"add Insert"*** e está disponível na barra de funções *Mold Base Components* do MTD.

Este comando executa Postiços num Molde com formas predefinidas, onde só se podem alterar as dimensões. O tipo de Postiços que faz são:

- ⇨ **Pad with Chamfer** (Postiço rectangular com chanfro)
- ⇨ **Pad with Fillet** (Postiço rectangular com boleado)
- ⇨ **Shaft** (Postiço cilíndrico)

O problema desta ferramenta é que as formas (rectangulares e circulares) não podem ser alteradas, ou executamos os postiços da forma predefinida ou então, se existir a necessidade de ter uma forma diferente, vamos ter de os modelar da forma normal. Existe a possibilidade de adicionar outras formas ao catálogo, mas serão sempre formas predefinidas.

A metodologia para inserir um Postiço será brevemente descrita para fazer um ponto de comparação com a aplicação anteriormente desenvolvida. Desta forma, vamos começar por clicar no comando "*add Insert*". O quadro da figura 77 vai-nos aparecer de seguida. Aqui vamos ter a possibilidade de escolher o tipo de Postiço se clicarmos no ícone para ter acesso ao catálogo (figura 78).

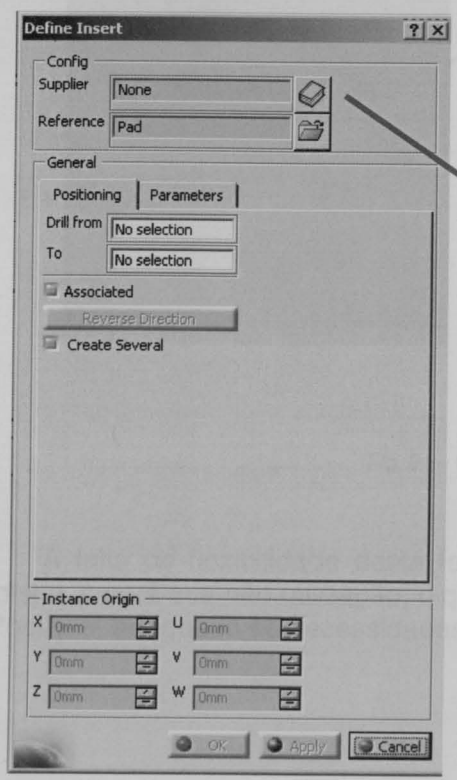


Fig. 77 – *Define Insert*

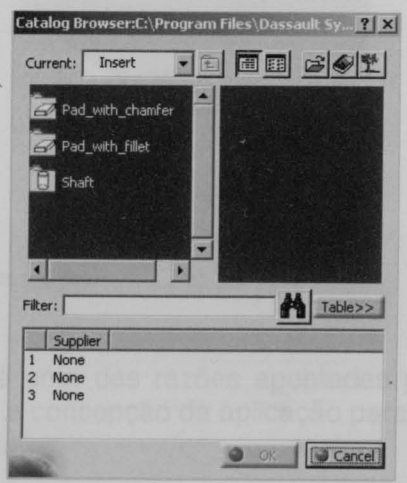


Fig. 78 – Catálogo do *Define Insert*

Depois de definir qual o tipo de Postiço que queremos, vamos indicar a partir de onde queremos abrir o espaço para o sólido criado na placa e até onde queremos furar. Estas indicações são dadas nos campos *"Drill From"* e *"To"* do quadro apresentado na da figura 77. Assim, vai-nos aparecer o Postiço preliminar, onde podemos alterar as dimensões e a sua posição (figura 79). Depois de pronto o Postiço que fica é o da figura 80, onde vamos depois ter e fazer um *Split Component*, à semelhança daquilo que fizemos quando concebemos a estrutura do Molde. O resultado final será o da figura 81.

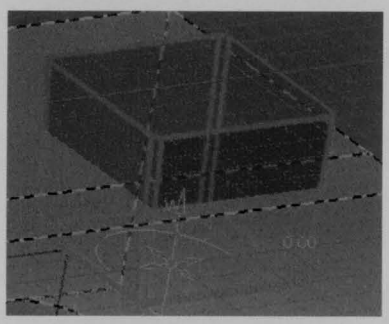


Fig. 79 – Postiço preliminar

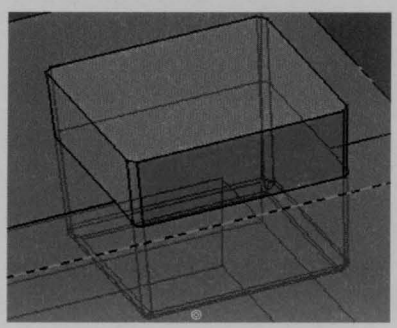


Fig. 80 – Postiço sem *Split*

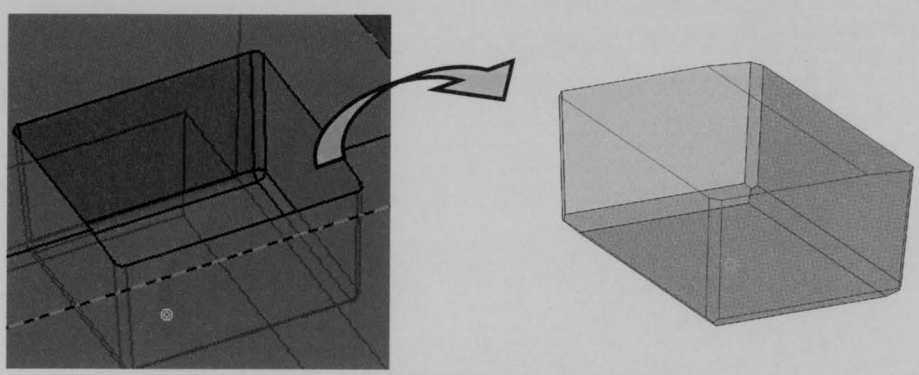


Fig. 81 – Caixa e Postiços prontos

A falta de flexibilidade desta ferramenta é uma das razões apontadas pelos Moldistas para a sua não utilização, o que justifica a concepção da aplicação para criar os Postiços, adequada às necessidades do cliente.

Controlo de Erros

10

10. Controlo de Erros

O controlo do erro do utilizador constitui um capítulo importante em qualquer aplicação que se destine a ser utilizada por várias pessoas. De facto, quando concebemos este tipo de aplicações temos de imaginar que o utilizador vai executar todo o tipo de operações que introduzam entradas erradas no decorrer do programa. Vamos começar por descrever os vários tipos de erros, para depois passar ao caso concreto das nossas aplicações.

10.1 Tipos de Erros

Na realização de aplicações deste tipo, há que ter atenção a três tipos diferentes de Erros:

⇨ Erros de Compilação:

Estes erros são os que se verificam quando desenvolvemos a aplicação. Os principais erros nesta categoria são os de sintaxe, os de tipos de variáveis e os de dados. No VBA a maioria destes erros são detectados e apontados, de imediato, no editor VBA.

⇨ Erros de Execução:

Estes erros, também designados por "*run-time*", ocorrem quando a aplicação é executada e se verificam situações não previstas pelo código. Como exemplos, temos a entrada incorrecta de dados por parte do utilizador, a tentativa de criação de um objecto com o mesmo nome de um já existente ou a abertura de um ficheiro não localizado correctamente ou que não exista. A correcção deste tipo de erros é muito mais complexa, já que o programador tem de prever o maior número de situações de erro possíveis, isto é, fazer uma aplicação que consiga controlar e orientar qualquer utilizador. Este tipo de erros vão ser os analisados no capítulo seguinte.

⇨ Erros Lógicos:

Estamos na presença de um erro deste tipo se na criação ou execução da aplicação não ocorrem erros, mas ainda assim, o resultado final não é o pretendido. Este tipo de erros são chamados "*Bugs*" e podem ser de muito difícil detecção, obrigando a uma cuidada análise da aplicação e a frequente necessidade de instalação de código temporário para testes.

Como vamos poder ver mais à frente, os erros de execução vão ser os mais importantes e aqueles que é necessário, a todo custo, controlar para evitar que o utilizador tenha acesso ao código.

10.2 Prevenção de Erros de Execução

Quando acontece um erro deste tipo, se o programador não prevenir nada em contrário, o CATIA mostra uma caixa de diálogo com a menção e o código do erro detectado e com a possibilidade de terminar a aplicação ou executar um *Debug* (figura 82). Se escolhermos o opção "*Debug*" o VBA leva-nos de volta ao código para nos indicar onde está o erro. É precisamente esta situação que temos que evitar a todo custo.

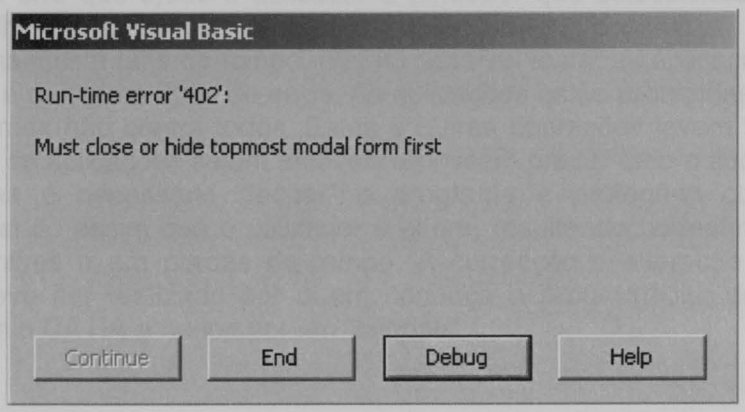


Fig. 82 – Caixa de dialogo quando acontece um erro

Quando concebemos uma aplicação deste tipo, por vezes só nos apercebemos da enorme quantidade de erros que o utilizador pode cometer, quando testamos a aplicação. Existem outro tipo de erros que são mais previsíveis e que são controlados logo à partida. Podemos começar por falar dos erros na introdução dos dados. Vamos imaginar que o utilizador em vez de introduzir o valor de uma altura, por engano introduz texto misturado com o número; esta situação vai dar origem automaticamente à abertura do código do programa e é um caso que pode ser facilmente controlado, colocando uma instrução ***If*** que declara que, se o utilizador preencheu mal algum campo ou se esqueceu de o preencher , ou mesmo a situação de introduzir um 0 em vez de um valor positivo ou negativo, aparecerá uma mensagem aconselhando à sua correcção.

Como já podemos verificar, o leque de erros possíveis parece não ter fim. Em qualquer uma das aplicações vamos ter de introduzir o nome da *Part* correctamente para que esta seja devidamente processada. Se o utilizador se enganar nesse nome também vamos encontrar problemas. Este tipo de erros podem ser resolvidos com algumas ferramentas fornecidas pelo Visual Basic. O VB inclui a instrução "*ON ERROR*" para prevenir este tipo de situações. Esta instrução permite definir o que se vai passar com a aplicação quando se verifica um erro de execução. Existem três possibilidades de aplicações: envio para uma zona predefinida onde se procede ao tratamento do erro, ignorar a instrução do erro e continuar a aplicação e cancelar a gestão de erros. Para o caso referido atrás, vamos utilizar a instrução "*On Error GoTo*". Esta instrução dispõe das três sintaxes seguintes:

- ⇨ **On Error Resume Next**
- ⇨ **On Error GoTo** *instrução*
- ⇨ **On Error GoTo 0**

Assim, vamos poder definir que em caso de erro aparecerá uma mensagem a avisar o utilizador que introduziu mal algum parâmetro.

Estes poderão ser os erros típicos associados ao menu inicial, pelo menos, foram os detectados até esta data. Vamos passar agora a falar em erros que ocorrem quando se corre o resto do programa. Por exemplo, dentro da aplicação Postiços quando estamos a fazer um *Fillet*, é-nos pedido para seleccionar uma aresta e vamos imaginar que o utilizador se engana e selecciona uma aresta, por exemplo, do Molde; é natural que ocorra um erro visto que a aresta que seleccionamos não pertence ao corpo no qual estamos a trabalhar. É necessário também aqui, introduzir uma mensagem de erro que ajude o utilizador a perceber que seleccionou mal uma das arestas e dar-lhe a possibilidade de corrigir a selecção.

Por manifesta falta de tempo, não foi possível testar as aplicações a fundo na procura deste e de outros tipos de erros. As aplicações estão protegidas contra alguns destes erros, mas não contra todos. Estas e outras operações levam tempo para se conseguir que as aplicações sejam estáveis e possam prever todo o tipo de falhas. De qualquer forma é necessário "fechar" o programa e protegê-lo contra qualquer alteração, evitando assim que o utilizador o altere, resultando normalmente em erros ainda mais graves e em perdas de tempo. A correcção e alteração deste tipo de aplicações, deve ser realizada por quem conheça a programação do código e se identifique com o CATIA e nunca por um "curioso".

Knowledgeware: Outra Forma de Automatizar o CATIA V5

11

11. Knowledgeware: Outra forma de Automatizar o CATIA V5

Para além do VBA, Visual Basic e VBScript, existe outra forma de Automatizar o CATIA V5. Esta outra forma existe embebida dentro do CATIA e chama-se ***Knowledgeware***. A função destes produtos é ajudar a empresa a consolidar o seus conhecimentos, normas internas, etc., dentro da própria concepção do produto final e no fundo, ajudar a tomar decisões de Engenharia para poder reduzir ao máximo os erros, e assim aumentar a produtividade. Para ter uma ideia do grau de personalização versus o grau de complexidade dos vários métodos para automatizar o CATIA, podemos ver o esquema da figura 83.

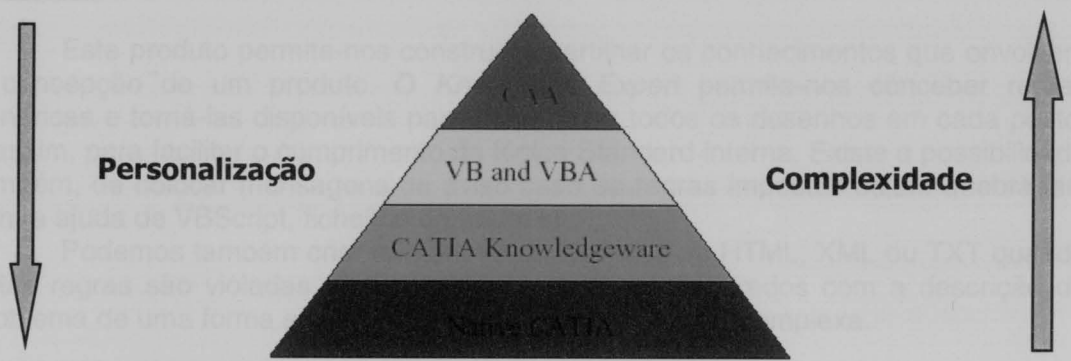


Fig. 83 – Comparação entre o Knowledgeware, VBA, VB e CAA

Para servir de termo de comparação, vamos falar das potencialidades de cada um dos produtos. Dentro desta *Workbench* podemos encontrar:

- ⇨ **Knowledge Advisor**
- ⇨ **Knowledge Expert**
- ⇨ **Product Engineering Optimizer**
- ⇨ **Product Knowledge Template**

Knowledge Avisor

Este produto dá-nos a possibilidade de colocar Regras, Parâmetros, *Checks* e Fórmulas dentro da concepção do modelo 3D para facilitar a verificação de condições que devem ser cumpridas na obtenção do produto final, de acordo com os requisitos do cliente. Para isto, o *Knowledge Advisor* oferece-nos as seguintes possibilidades:

- Automatizar a definição do produto
- Partilhar o *Know-how* com todos os utilizadores
- Aumentar a produtividade
- Concepção de componentes para posterior reutilização
- Guiar e Assistir os utilizadores durante as tarefas de desenho

Um breve exemplo daquilo que o *Knowledge Advisor* é capaz, é por exemplo a concepção de tabelas de desenho para a concepção de componentes normalizados exclusivos da empresa; assim, podemos conceber um componente com os respectivos parâmetros e respectivas fórmulas ou regras, e consolidar toda essa informação numa tabela de desenho que nos vai dar a possibilidade de, mudando alguns parâmetros, não desrespeitar nunca as regras impostas à partida. Vamos imaginar que uma peça não pode ultrapassar uma certa massa; podemos colocar Parâmetros que nos definam a massa total do componente e o tipo de material. Este poderá ser outro exemplo simples daquilo que é possível implementar.

Knowledge Expert

Este produto permite-nos construir e partilhar os conhecimentos que envolvem a concepção de um produto. O *Knowledge Expert* permite-nos conceber regras genéricas e torná-las disponíveis para monitorizar todos os desenhos em cada posto, e assim, para facilitar o cumprimento da lógica Standard interna. Existe a possibilidade também, de colocar mensagens de aviso caso as regras impostas sejam quebradas, com a ajuda de VBScript, ficheiros de texto, etc.

Podemos também criar relatórios nos formatos de HTML, XML ou TXT quando estas regras são violadas. Estes relatórios podem ser gerados com a descrição do problema de uma forma simplificada ou de uma forma mais complexa.

Product Engineering Optimizer

Em Engenharia, a concepção de um componente pode passar por um processo iterativo que por vezes, se pode tornar muito complexo. O problema fundamental existente no desenvolvimento de um componente é a optimização de um conjunto de variáveis. O que o *Product Engineer Optimizer* nos permite é conceber essa optimização dentro do próprio modelo 3D. Para este efeito podemos definir as variáveis a ser optimizadas e introduzir regras, fórmulas, *loops*, etc., e no fim obter um produto optimizado.

Product Knowledge Template

Este produto permite-nos a captação de toda a metodologia envolvida na concepção de componentes, não só da geometria mas também de todo o conjunto de parâmetros e regras que envolvem essa concepção, e guardar toda esta informação para sua posterior utilização. Toda esta metodologia pode ser guardada em catálogos e torná-los disponíveis a qualquer utilizador dentro da empresa. Dentro deste produto também nos é oferecida a possibilidade de guardar essa informação na forma de *Macros*, utilizando uma linguagem de VBScrpting interna do *Knowledgeware*. Para além disto, podemos bloquear o acesso, de outras pessoas externas à empresa, a toda a metodologia da empresa.

Conclusão e Desenvolvimentos Futuros

12

12. Conclusão e Desenvolvimentos Futuros

12.1 Conclusão

O objectivo deste trabalho era estudar as diversas formas de personalizar uma ferramenta CAD/CAM seguindo os requisitos do cliente e da própria Indústria de Moldes, sendo que a ferramenta em causa é um dos softwares de modelação 3D mais utilizados na concepção de componentes para a Indústria Automóvel, o CATIA V5.

Com este trabalho, chega-se à conclusão que existem muitas tarefas que podem, de facto, ser agrupadas numa aplicação, permitindo diminuir o tempo necessário para as realizar, sendo que, algumas tarefas são de um grau de complexidade elevado e outras são, pelo menos à partida, mais simples. Para a concepção de uma aplicação deste tipo é necessário estudar o problema a em pormenor, visto que, o que à partida parece ser fácil, em programação, pode tornar-se muito difícil e complexo. É importante realçar que, antes do início da realização deste trabalho os conhecimentos sobre o Visual Basic, VBA ou VBScript eram nulos, por isso foi necessário, numa primeira fase e mesmo antes de iniciar o estágio curricular, efectuar uma aprendizagem exaustiva desta linguagem de programação através da leitura de livros e dos manuais da cadeira de Programação de Computadores da Licenciatura em Engenharia Mecânica da Faculdade de Engenharia da Universidade do Porto, secção de Matemática. Ainda foram concebidas algumas aplicações muito simples em AutoCAD 2002, com o objectivo de conhecer uma API que trabalha também com elementos gráficos, embora diferentes dos utilizados em CATIA V5, até porque num software estamos a trabalhar em 2D e noutro estamos a modelar em 3D.

O início deste estágio curricular deu-se com a frequência de um curso de formação dado pela CadTech Ibérica, S. A., onde foram obtidos conhecimentos relativamente ao CATA V5, entre os quais, a metodologia de modelação em 3D. Ainda, a frequência de outro curso de formação em Automação do CATIA V5 realizado pela IBM, foi fundamental para a abordagem da API deste software e assim conjugar todos os conhecimentos adquiridos até aqui para conceber as aplicações em causa.

O conhecimento do CATIA V5 tornou-se fundamental para o desenvolvimento destas aplicações, visto que, só conhecendo as virtudes e limitações do software em causa, se podem conceber estratégias de concepção automática de componentes e que contenham a metodologia mais correcta de forma a minimizar ao máximo a complexidade e o tempo necessário para os desenvolver. A leitura atenta de toda a documentação disponível relativa à API do CATIA V5, representou uma fase importante nesta aprendizagem, nomeadamente com a execução de exemplos concretos.

Relativamente às aplicações concebidas, podemos realçar o facto de estas terem sido criadas de forma a conservar uma estrutura correcta em termos de metodologia CATIA V5, e ao mesmo tempo serem o mais flexíveis, dentro do possível, permitindo ao utilizador realizar a tarefa pretendida, ou seja, fazer Eléctrodos com a forma que quiser ou fazer Postiços com uma forma qualquer, com boleados e chanfros onde for necessário ou mesmo sem eles. Ainda relativamente ao controlo de erros, podemos dizer que esta aplicação está prevenida contra alguns erros, como foi explicado no capítulo anterior, quando o erro é detectado não existe a possibilidade de recuar automaticamente naquilo que fizemos; assim, é necessário sair da aplicação para desfazer aquilo que a aplicação realizou e voltar a fazer tudo de novo.

Falta ainda referir, a possibilidade que existe de criar uma nova barra de ferramentas que pode ter o título de *Macros* e nesta barra, colocar sob a forma de ícones o arranque das nossas aplicações, sendo muito mais directo aceder às aplicações do que ir ao menu e carregar a aplicação sempre que se pretenda fazer uso dela.

Concluindo, penso que este trabalho servirá para abrir os horizontes numa área que tem sido pouco explorada, em termos de Moldes e tudo o que está associado a este tipo de Indústria. Este tipo de aplicações, devidamente testadas e optimizadas, podem tornar-se ferramentas de grande utilidade e que pode melhorar consideravelmente a produtividade e fiabilidade na concepção de Moldes para todo tipo de materiais. Durante o decorrer deste trabalho existiu a dificuldade de adquirir conhecimentos dirigidos à execução do projecto, que ocuparam uma grande parte do tempo de duração previsto para este estágio curricular. É por esta razão que as aplicações funcionam mas ainda não são estáveis, visto que será necessário tempo para realizar mais testes e conseguir o objectivo final que será a sua comercialização. Não posso deixar de dizer que foi um trabalho difícil, mas ao mesmo tempo aliciante visto que estive a explorar um campo que quase ou nada tinha explorado, pelo menos em Portugal, mesmo tendo em conta que a programação é um trabalho ingrato, e que por vezes, as pessoas que vão usufruir dele não imaginam todo o esforço que está por trás de coisas consideradas simples.

12.2 Desenvolvimentos Futuros

Para além das aplicações concebidas, existe um horizonte de aplicações, que podem não limitar-se única e exclusivamente à Indústria de Moldes, e que devem passar por um estudo cuidado das vantagens e desvantagens que uma aplicação destas trás à modelação e concepção de um componente.

Como projectos futuros podemos destacar, a inversão do sentido de eixos do Molde de acordo com os requisitos da indústria de Moldes da Marinha Grande, projecto este que já foi referido no capítulo 8. Para além deste projecto, existe um muito mais ambicioso e muito mais complexo que todos aqueles de que já falamos até agora e que corresponde a conceber um Molde "Inteligente", em que a respectiva estrutura se adapte às dimensões do Postiço ou ao atravancamento da peça. Vamos imaginar que o fabricante de Moldes utiliza sempre o mesmo tipo de estrutura e que esta estrutura não é comprada, ou seja, ela é fabricada dentro da empresa e se rege por dimensões estipuladas pelas regras internas da empresa. Vamos imaginar que temos uma peça que já se encontra dentro de um Postiço e relativamente ao qual já se fez à separação Macho / Cavidade (figura 84).

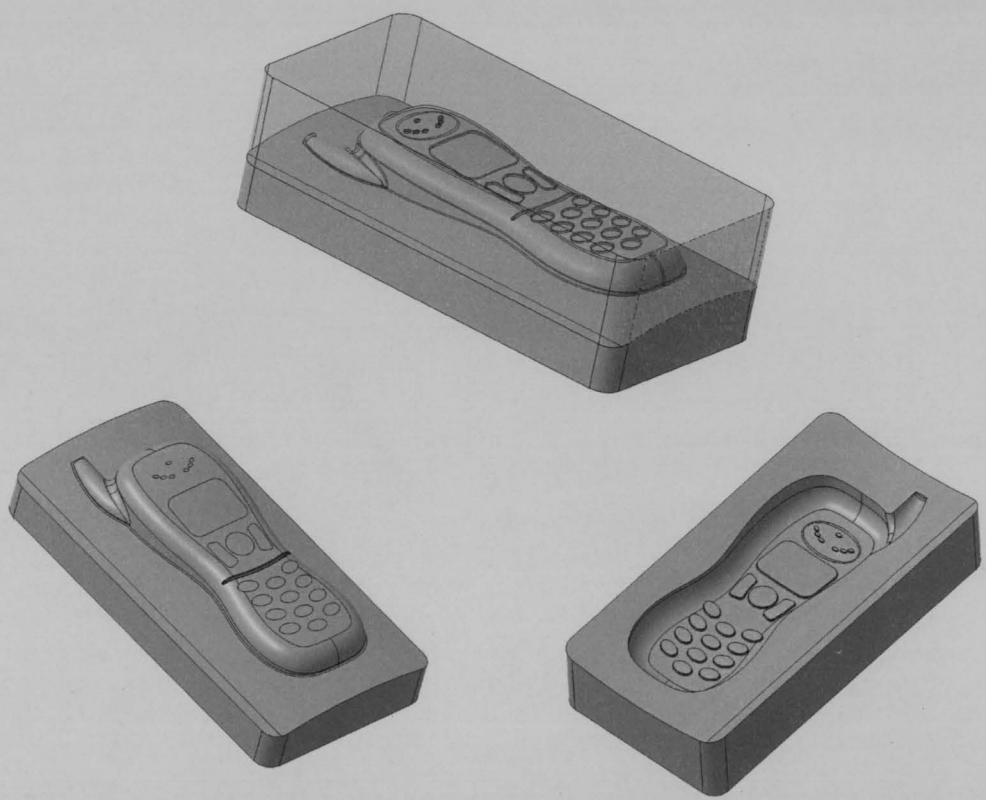


Fig. 84 – Postiços para os quais se pretende uma estrutura

A ideia é conceber uma aplicação que, perante as medidas máximas do Postiço, seja o suficientemente "inteligente" para escolher uma estrutura que lhe seja a adequada, respeitando as distâncias definidas pelo cliente. A ideia aqui não é ir contra a metodologia do MTD, mas sim aproveitar todas as suas potencialidades apenas agrupando e implementando as tarefas para estabelecer as dimensões das placas (comprimento, largura, espessura, etc.).

Em vez de estabelecer manualmente, todas as dimensões das placas em relação ao atravancamento do Postiço, esta operação será feita automaticamente, recorrendo talvez a fórmulas ou regras que a partir das dimensões máximas possam recalcular as dimensões do Molde (figura 85). Este projecto, embora aliciante, envolve algoritmos muito complexos (prever regras internas da empresa, dimensões máximas e mínimas, sistema de injecção, etc.) e que demorariam muito tempo a conceber e implementar, para além de não existir a garantia de que, depois de concebida a estrutura do Molde, o MTD a consiga absorver aceitando que os sólidos resultantes desta operação sejam de facto sólidos do MTD.

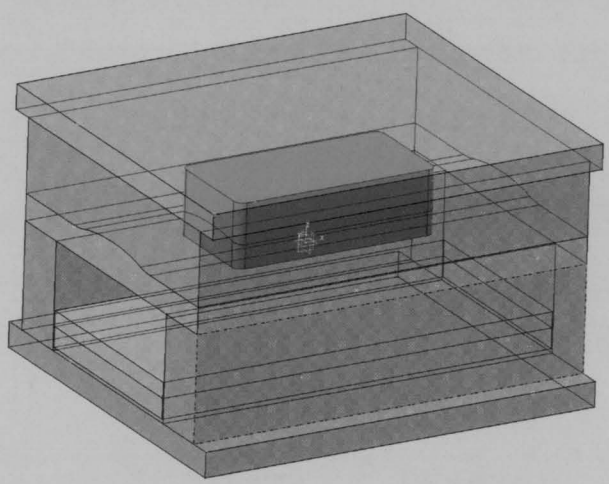


Fig. 85 – Estrutura do Molde ajustada ao Postiço

Bibliografia

13

13. Bibliografia

- FONSECA, Joaquim O. - Apontamentos da Cadeira de Fabrico de Moldes, Opção Tecnologias de Moldação por Injecção - Licenciatura em Engenharia Mecânica, FEUP, 2003.
- FONSECA, Joaquim O. – Apontamentos da Cadeira de Concepção e Fabrico Assistido por Computador, Licenciatura em Engenharia Mecânica, FEUP, 2002.
- MENGES, G. e MOHREN,P. - How to Make Injection Molds, 2nd ed. Munich: Hanser, 1993.
- PYE, R. G. W. - Injection Mould Design. London: Edit. George Godwin Limited, 4ª Ed.
- GASTROW - Injection Molds, 130 Proven Designs.
- CRUZ, Sérgio - Moldes de Injecção de Termoplásticos: Hemus.
- PUGA, André T. - Apontamentos da cadeira de Programação de Computadores, Licenciatura em Engenharia Mecânica, FEUP, 2003.
- DASSAULT SYSTÈMES – Mold Tooling Design Fundamentals, Version 5 Release 10, 2003.
- DASSAULT SYSTÈMES – Mold Tooling Design User Component Management, Version 5 Release 10, 2003.
- DASSAULT SYSTÈMES – Introduction to the Mathematical Concepts of CATIA V5, Version 5 Release 10, 2003.
- DASSAULT SYSTÈMES – CATIA V5 Automation, Version 5 Release 10, 2003.
- NINA, Nuno - Visual Basic 6: Curso Completo, FCA Editora de Informática, Lisboa, 1999.
- CAMPOS, L.; VILAR, S.; LÚCIO, L. – Programação em Visual Basic 6, FCA Editora de Informática, Lisboa, 2001.
- BALENA, Francisco – Programing Visual Basic 6.0, Redmond: Microsoft Press, cop. 1999.
- HALVORSON, Michaël – Microsoft Visual Basic 6.0: Passo a Passo, McGraw-Hill, Lisboa, 1999.
- MICROSOFT CORPORATION – Microsoft Mastering: Microsoft Visual Basic 6.0 Fundamentals, Redmond: Microsoft Press, cop. 1998.

- MICROSOFT CORPORATION – Microsoft Visual Basic 6.0: Programmer's Guide, Redmond: Microsoft Press, cop. 1998.
- HOLZNER, Steven – Visual Basic 6 Black Book, The Cortolis Group, 1998.
- FERREIRA, Luís e SANTOS, João - Programação em AutoCAD - Curso completo, FCA Editora de Informática, 2002.
- Página pessoal J. O. Fonseca: *www.fe.up.pt/~fonseca*
- Secção de Matemática – FEUP: *www.fe.up.pt/smat/Programação.htm*
- COE Forum: *www.coe.org*
- CAA V5 Community: *www.caav5.com*
- Dassault Systèmes: *www.3ds.com*
- Dassault Systèmes CATIA V5: *www.3ds.com/en/brands/catia_ipf.asp*
- CATIA Digital Digest: *http://d-digest.com/catiadigitaldigest*
- CATIA Tips: *www.catiatips.de*
- International Private CATIA V5 User Board: *www.catiav5forum.de*
- MSDN – Microsoft Visual Basic: *http://msdn.microsoft.com/vbasic*

prodep III
Mais Educação

UNIÃO EUROPEIA
Fundo Social Europeu

Nome: Nelson Luís Vieira Pinto
Curso: Eng. Mecânica
Datas: 2/17/2003 a 5/16/2003
Tema: Optimização e personalização de uma ferramenta de CAD/CAM de acordo com os requisitos próprios da industria de moldes
Empresa: CADTECA
Concurso: 306/012-03 – PRODEPII – Medida 3/Acção 3.2 - Estágios