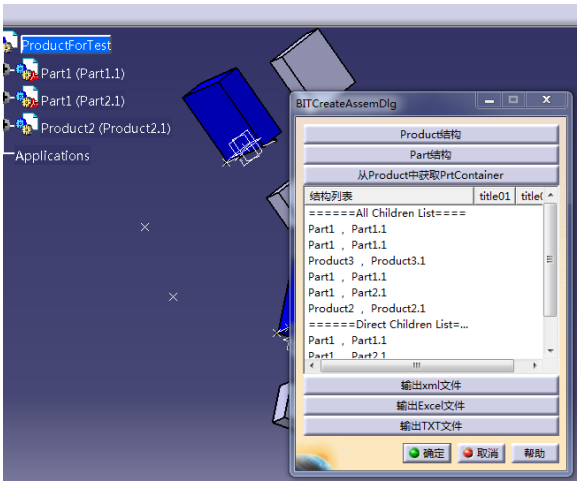
第 4 讲 CATIA 文档结构和操作

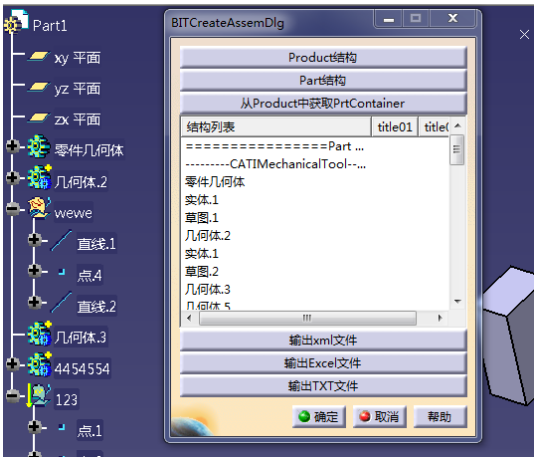
目录

1. 概述

目标 1.在 CATProduct 环境下获得 Product 的组件和零件的名称，并将其显示在对话框里。



目标 2.在 CATPart 环境下获得文档内的内容，并将其显示在对话框里。如图：



目标 3：在 CATProduct 环境下，获得所有的 Part 中特征。如图：

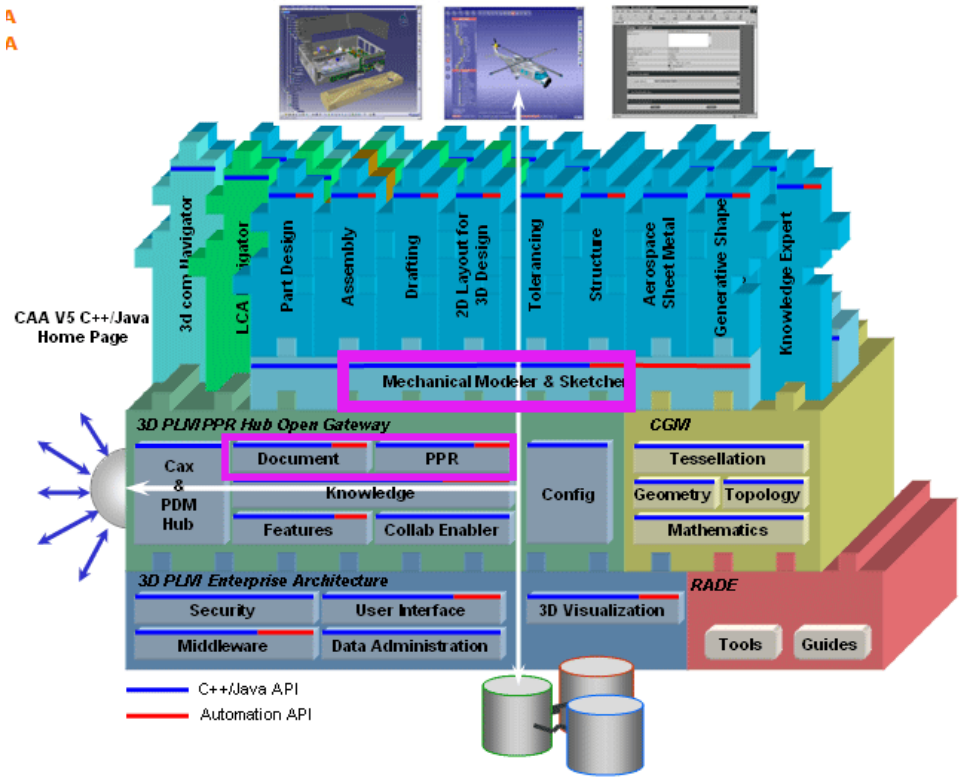
1.1 简介

百科全书->Document->file 选项卡里边的 Document OverView

百科全书->PPR->Product 选项卡里边的所有 Technical Article 和 Use Cases：Working with Product Structures 与 Working with Products

百科全书->Mechanical Modeler&Sketch->Use Case ： The Part document

如下图所示：百科全书中粉红色框起来的部分



1.2 界面布局

这部分内容用到 Train1 中所完成内容。

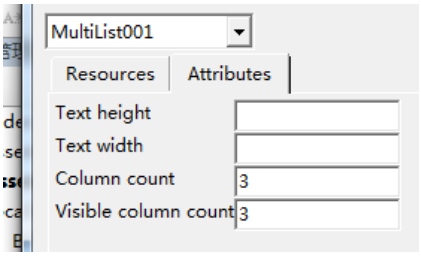
以 Train1 中创建的对话框为基础，在对话框中添加 MultiList 控件即可：

打开 BITCreateAssemDlg.CATDlg

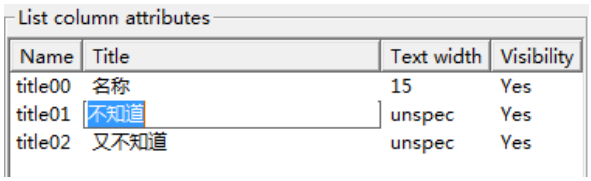
添加控件 MultiList，并修改属性，如下图所示，MultiList Name 为 multiList001

添加字段“名称”“不知道”“又不知道”将名称栏的长度调得稍微大点。

先修改 Attribute 中的 Column count 为 3。点击应用再切到 resource 换选项卡，不点击应用不可以。



在 List column attribute 中修改 Title：先左键选中 title 列的一个单元格，再点击该单元格便是修改。注意此处不是双击。Text width 是指列的宽度，可以将名称列设置为 15。



修改过后的的 Dialog 对话框如下：

2. 装配体（Product）文档

2.1 装配体（Product）文档结构

一个 Product 是由不同的 components 组成，当设计一个 Product 的时候，首先必须确定都有哪些 components 组成，然后依次设计 component。产品结构框架容许不同的 component 装配在一起形成最终产品，并以产品结构树的形式体现出来。

为了让一个 CATPart 文件和 CATProduct 文件能够装配在一起，和在不同的装配体重重复使用，每个 part 和 Product 都有一个具体的 root Product 进行管理装配情况。所以创建一个装配结构必须的步骤为：

1.创建一个 CATProduct 文件；

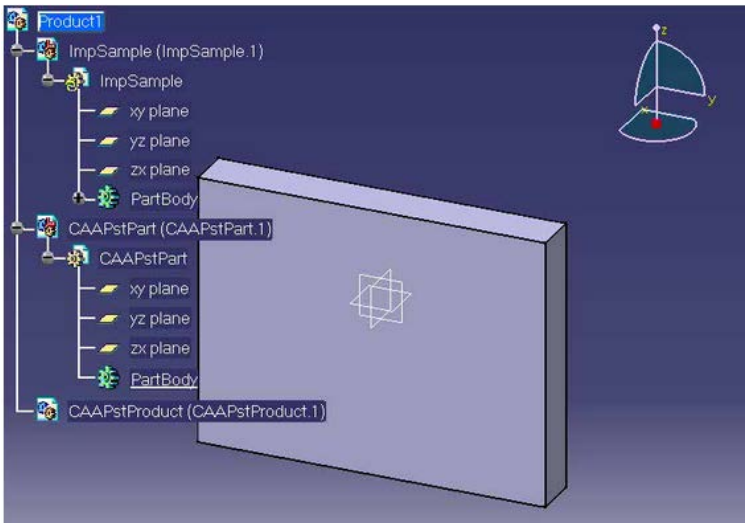
2. 获取这个文件的 Root Product 元素；

3.添加新的 components 到 root Product；

在 CATIAV5 版本中只有 CATPart 文件和 CATProduct 文件有 root Product 元素，这就意味着只有这两个文件可以用来装配。导入装配的 component 称为参考。一个 part 或 Product 在一个装配体结构中由两个名称来定义：

一个是参考名称，或者称为 Part Number,这是这个 root product 参考文档的名称。

另外一个是产品实例名称，这是在一个文档作为组件导入到装配体中的唯一标示。



2.2 装配体（Product）文档遍历

每个 Product 节点都有它自己的子集，这些子节点也都是 Product 类型。所以，只要获取了一个 Product 的根节点，就可以获得它的其他子节点。获取根节点时一般首先获得 document，然后获得 CATIDocRoots 接口，再通过 CATListValCATBaseUnknown 获取一个列表中的元素，第一个列表中的元素就是根节点的方法：

```
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
CATDocument *pDoc = pEditor->GetDocument();
CATIDocRoots* piDocRootsOnDoc = NULL;
HRESULT rc = pDoc->QueryInterface(IID_CATIDocRoots,
                                  (void**) &piDocRootsOnDoc);
/////获得Product的Root，是第一个元素
CATListValCATBaseUnknown_var* pRootProducts = piDocRootsOnDoc->GiveDocRoots();
CATIProduct_var spRootProduct = NULL_var;
if (pRootProducts && pRootProducts->Size())          //如果pRootProduct不为空，
获得第一个元素
{
    ShowText("pRootProducts", pRootProducts->Size());
    spRootProduct = (*pRootProducts)[1];
```

```
    delete pRootProducts;
    pRootProducts = NULL;
}
piDocRootsOnDoc->Release();
piDocRootsOnDoc = NULL;
////// Get CATIProduct handle on the root product.
CATIProduct *piProductOnRoot = NULL;
rc = spRootProduct->QueryInterface(IID_CATIProduct,
                                   (void**) &piProductOnRoot);
```

获取其他所有节点的方法：CATIProduct::GetAllChildren(“CATIProduct”);

获取其他直接节点的方法：CATIProduct::GetChildren(“CATIProduct”);

获取 Product 实例名称的方法：CATIProduct::GetPrdInstanceName();

获取 Product 名称的方法：CATIProduct::GetPartNumber();

给 Product 在添加一个 Product 的方法：CATIProduct::AddProduct ();

删除一个 Product 节点的方法：CATIProduct::RemoveProduct ();

还有其他的操作方法如修改实例名称，修改节点的位置，修改 PartNumb 等。

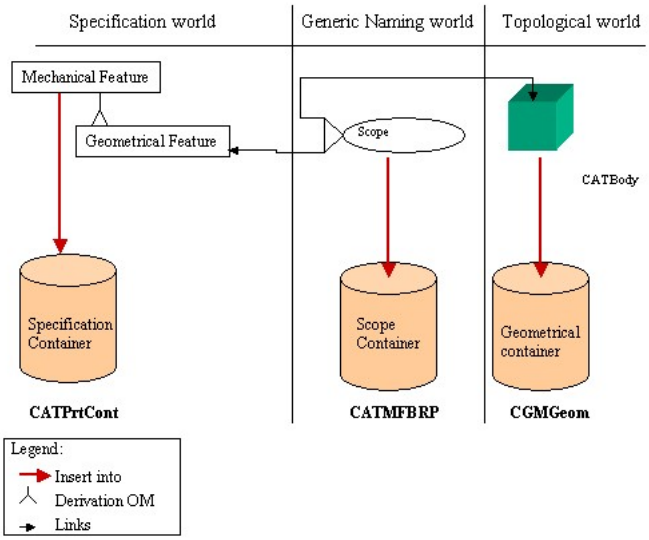
3. 零件（Part）文档结构

一个 part 文档包含有固定数目的 containers，如下图所示：

这四个 container 分别是：产品 container（CATProdCont），结构 container（CATPrtCont），scope container（CATMFBRP）和几何 container（CGMGeom）。这四个 container 互相联系共同构成了一个 Part 文档。

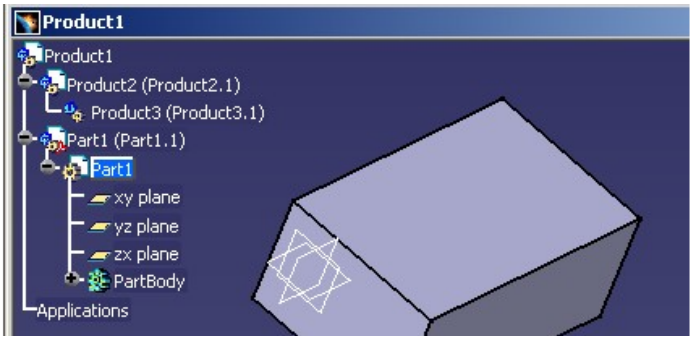
3.1 不同 Container 之间的关系

所有的计息特征都包含在机构 container 中



3.2 Product Container（CATProdCont）

任何 Part 文件都有一个产品 container（CATProdCont），它包含了一个 ASMProduct 特征，这个特征是这个 Part 文件在产品文件中的实例的链接。如图所示 Part1.1 就是 Part1 文件的 ASMProduct 特征的一个实例。



产品实例（Part1.1）包含了 Part 在 Product 中的位置信息。

3.2.1 从 Product 中获取产品 container

通过 CATPrtPart 接口可以获得 ASMProduct 特征。从一个 Product 实例中获取 Part 文档采用的是 CATILinableObject 接口，如下边代码所示。ispProduct 就是一个 Product 实例。

```
CATIProduct_varspRef = ispProduct->GetReferenceProduct();
      if ( NULL_var != spRef )
   {
CATILinkableObject * piLinkableObject = NULL;
rc = spRef->QueryInterface( IID_CATILinkableObject, (void**)&piLinkableObject );
        if ( SUCCEEDED(rc) )
        {
            // Do not release the document pointer
CATDocument * pDocument = NULL ;
pDocument = piLinkableObject->GetDocument();//从 Product 的文档中获得 Part 文档
```

3.2.2 从 Part 中获取产品 container

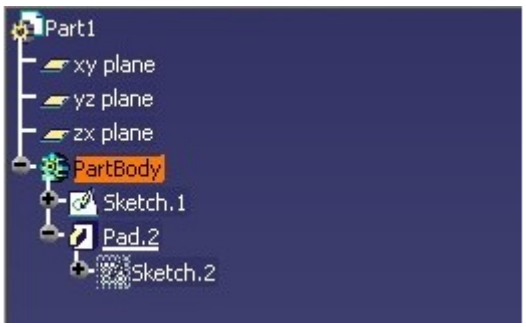
获得了产品 container（CATProdCont）的代码如下：

```
CATIContainerOfDocument * pIContainerOfDocumentOnDoc = NULL ;
   HRESULT rc = pDocument->QueryInterface(IID_CATIContainerOfDocument,
                                          (void**)&pIContainerOfDocumentOnDoc)
   if (SUCCEEDED(rc) )
   {
CATIContainer * pIContainer = NULL ;
rc = pIContainerOfDocumentOnDoc->GetProductContainer(pIContainer);
      }
```

3.3 Specification container（CATPrtCont）

Specification container（CATPrtCont）包含了设计的机械对象，机械对象包含的设计要素如图中结构树所示。

建模过程和结果全部在结构 container 中，它的一部分内容具体表现为结构树



图中 Part1, xy plane, yz plane, zx plane, PartBody, Sketch and Pad.2 是机械特征，有三种机械特征。分别是 Part 特征，几何图形集，几何图形（CATBody）。

Part 特征是 part 中最主要的特征，它包含所有的设计特征；

几何图形集包含其他设计特征和其他几何图形集；

几何图形是设计特征的拓扑结果。

有三种方法可以获得结构 container（CATPrtCont）。

第一种是从 Part 文件的 CATIContainerOfDocument (MecModInterfaces)获取。代码为：

```
CATIContainer * pIContainer = NULL ;
rc = pIContainerOfDocumentOnDoc->GetSpecContainer(pIContainer);
```

第二种是 CATInit (ObjectModelerBase)，这个接口所有的 CATIA 文件都有。

对于一个 Part 文件，Root Container 是 CATPrtContainer，实现代码如下：

…

```
CATInit * pInitOnDoc = NULL ;
   HRESULT rc = pDocument->QueryInterface(IID_CATInit, (void**)&pInitOnDoc ) ;
   if ( SUCCEEDED(rc) )
   {
CATIPrtContainer *pIPrtCont = NULL ;
pIPrtCont = (CATIPrtContainer*)
pInitOnDoc->GetRootContainer("CATIPrtContainer");
   }

      ...
```

第三种是采用所有特征的 CATISpecObject 接口。

```
      ...
CATISpecObject * pSpecObjectOnFeat = NULL ;
   HRESULT rc = pMyFeat->QueryInterface(IID_CATISpecObject, (void**)&pSpecObjectOnFeat ) ;
   if ( SUCCEEDED(rc) )
   {
CATIContainer_varspISpecCont = pSpecObjectOnFeat->GetFeatContainer();
   }

      ...
```

这儿的 pMyFeat 是任何特征的指针，spISpecCont 就是 CATPrtContainer。

3.4 Scope Container（CATMFBRP）

The scope container which contains the objects necessary to access the sub-elements of the topology.

再设计中 Scope Container 常用来选择一个边，面，这些拓扑对象是机械特征的输入，Scope Container 包含拓扑要素。

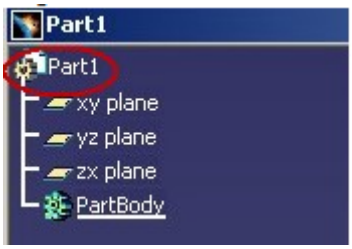
3.5 Geometrical Container（CGMGeom）

The geometrical container which contains the topological results of the mechanical features.

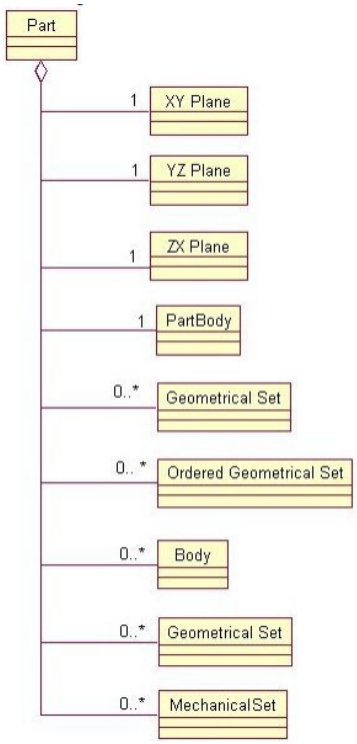
几何 Container（CGMGeom）包含几何特征的拓扑表示结果。

4. Specification container 中的 Part 特征

Part 特征是 Part 文档的最高级的特征。



Part 特征包含所有的设计对象，是所有 Part 文件机械特征的综合。它的内容如下图所示：



首先是 xy，yz，zx 三个参考平面。

其次是几何特征集：零件几何体，几何体，图形集，有序图形集。零件几何体（混合体）是最主要的集合，它在 Part 文件创建的时候自动产生并不能删除，为了提高设计的效率，创建了几何体，图形集，有序几何图形。

最后是机械集，这个集合主要由非几何特征集构成。

4.1 获取 part 特征

Part 特征一般从 CATIPrtContainer 接口的一个实例 获得。

...

CATISpecObject_varspSpecObjectOnPart = pIPrtCont->GetPart();

...

其中 pIPrtCont 是指向结构 container 的一个指针，

4.2 获取当前工作对象

当前对象就是用右键菜单中的定义“当前工作对象”命令后的特征对象。可以采用 CATIPrtPart 接口来获取当前特征。获取当前工作对象的方法：

CATIPrtPart ::GetCurrentFeature()

4.3 获取参考平面

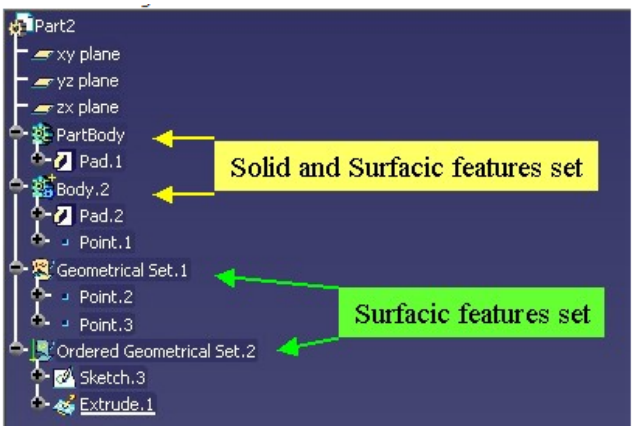
一旦获得了 Part 特征的指针，就可以用 CATIPrtPart 接口获取参考平面。代码如下：

```
...
CATIPrtPart * pIPrtPart = NULL ;
spSpecObjectOnPart->QueryInterface(IID_CATIPrtPart,(void**) &pIPrtPart ) ;
CATListValCATISpecObject_varListRefPlanes = pIPrtPart ->GetReferencePlane();
...
```

5. Specification Container 中的几何特征集

5.1 几何特征集

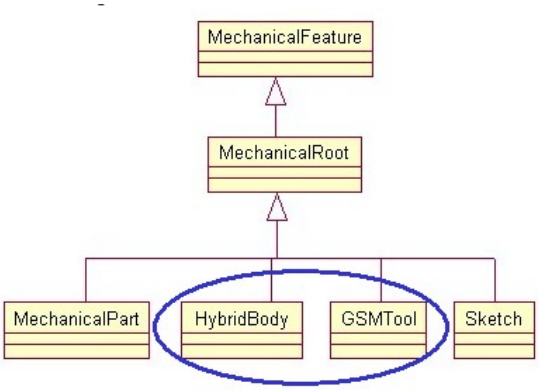
几何特征集包含了机械特征和其他的几何特征集们，有两种几何特征集：一是实体和曲面混合的几何特征集（PartBody，零件几何体、Body，几何体），二是只有曲面特征的几何特征集（Ordered Geometrical Set，OGS，有序几何特征集、Geometrical Set，GS，几何特征集）。如图所示：



所有的几何特征集都由 MechanicalRootStartUp 的一个 StartUp 实例创建。实体和曲面特征集（几何体、零件几何体）是由 HybridBody StartUp 创建。

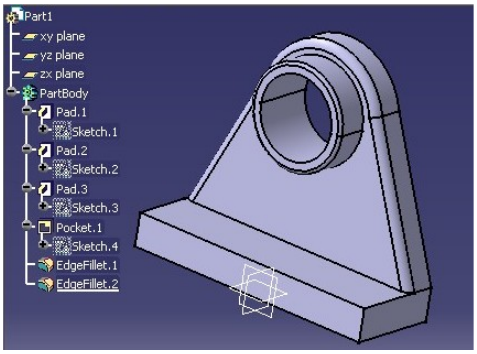
曲面特征集是由 GSMTool StartUp 创建。

MechanicalRoot 的架构如图所示。

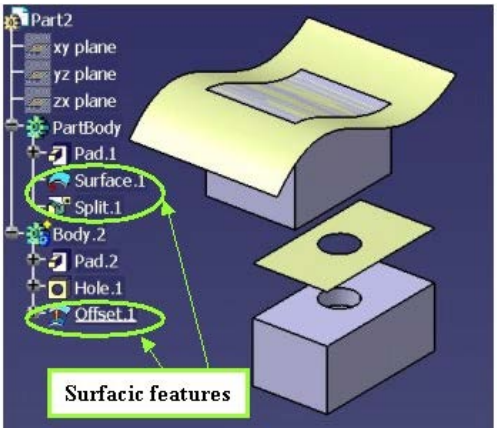


5.2 实体特征集（body）

几何体特征占用了一些体空间，可以为真实世界的对象建模，并能显示重力，重量，材料等属性。一个几何体特征如下图所示：



（实体和曲面特征集）几何体也可以包含点、线、面和曲面特征。如下图所示。



5.2.1 创建几何体特征集

在一个 part 文档中只有在有序几何图形集合特征和 Part 特征下才可以创建几何体。

用 CATIMechanicalRootFactoty 接口的方法 CreatePRTTool 可以 创建一个几何体特征。创建的方法为：

...

virtualCATISpecObject_varCreatePRTTool(constCATUnicodeString&iUserName,

constCATISpecObject_var&iDestination) = 0 ;

5.2.2 获取零件几何体（PartBody）

零件几何体在 Part 文档创建的时候自动创建并且不能删除，采用 CATIPartRequest 接口既可以获取零件几何体。代码如下：

...

```
CATIPartRequest *pICATIPartRequestOnPart =NULL ;
   HRESULT rc = spSpecObjectOnPart->QueryInterface(IID_CATIPartRequest,
                                          (void**)&pICATIPartRequestOnPart );
if ( SUCCEEDED(rc) )
   {
CATBaseUnknown_varThePartBody ;
CATUnicodeStringViewContext="MfDefault3DView";
rc = pICATIPartRequestOnPart->GetMainBody(ViewContext,ThePartBody);
   ...
```

其中 spSpecObjectOnPart 是指向 Part 特征的一个智能指针。

5.2.3 获取几何体（Body）

使用 CATIDescendants 接口即可获取所有的 Part 文档中的几何体特征。代码如下：

...

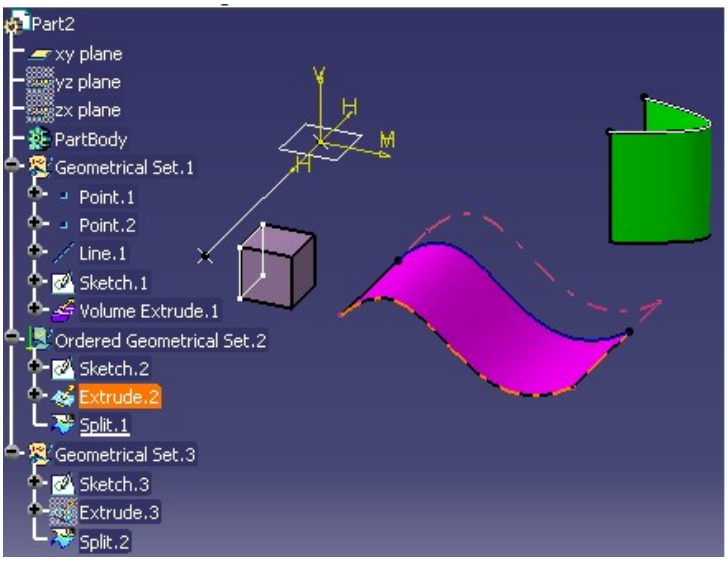
```
CATIDescendants *pIDescendantsOnPart =NULL ;
   HRESULT rc = spSpecObjectOnPart->QueryInterface(IID_CATIDescendants,
                                                   (void**)&pIDescendantsOnPart);
if ( SUCCEEDED(rc) )
   {
CATLISTV(CATISpecObject_var) ListResult ;
rc = pIDescendantsOnPart ->GetAllChildren("CATIMechanicalTool",
ListResult);
```

...

采用 GetAllChildren 方法能够获得所有的几何体特征。

5.3 曲面特征集

曲面特征集主要是由无厚度的面、曲面、线框几何元素构成，也可以包含体特征。



5.3.1 创建曲面特征集

曲面特征集分为有序几何图形集（OGS）和几何图形集（GS），它们所在的位置如图所示：

	Part	GS	OGS	Body
GS				
OGS				

涂红表示可以创建。

创建几何图形集的方法 CreateGeometricalSet 为：

```
virtual HRESULT CreateGeometricalSet(constCATUnicodeString&iUserName,
constCATISpecObject_var&iDestination,
CATISpecObject_var&oGeomSet,
intiPosition=-1 ) = 0 ;
```

创建有序几何图形集的方法为：CreateOrderedGeometricalSet ；

5.3.2 获取曲面特征集

有两种方法可以获取曲面特征集。

一种是不用特征集的名称，采用 CATIDescendants 接口。代码如下：

...

```
CATIDescendants *pIDescendantsOnPart =NULL ;
   HRESULT rc = spSpecObjectOnPart->QueryInterface(IID_CATIDescendants,
                                                   (void**)&pIDescendantsOnPart);
if ( SUCCEEDED(rc) )
```

```
  {
CATLISTV(CATISpecObject_var) ListResult ;
rc = pIDescendantsOnPart ->GetDirectChildren("CATIMmiOrderedGeometricalSet",
ListResult);
  ...
```

在这里获取了所有的在 Part 特征下的有序几何图形集。

采用 CATIMmiOrderedGeometricalSet 可以获得有序几何图形集，采用 CATIMmiNonOrderedGeometricalSet 可以获得几何图形集，采用 CATIMmiGeometricalSet 可以两着兼得。

第二种方法是采用特征集的名称，用 CATIPartRequest 接口获得。代码如下：

```
...
CATIPartRequest *pICATIPartRequestOnPart = NLLL ;
  HRESULT rc = spSpecObjectOnPart->QueryInterface(IID_CATIPartRequest,
                                                  (void**)&pICATIPartRequestOnPart );
if ( SUCCEEDED(rc) )
  {
CATListValCATBaseUnknown_varListSurfBodies ;
CATUnicodeStringViewContext ="MfDefault3DView";
rc = pICATIPartRequestOnPart->GetSurfBodies(ViewContext,ListSurfBodies);
  ...
```

这儿的字符串 ViewContext 就是图形集显示名称。

6. 遍历装配体（Product）树的实现

6.1 添加遍历代码

打开 BITCreateAssemDlg.cpp，找到

```
void BITCreateAssemDlg::OnSelectObjFstBtnPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{}
```

在大括号中添加：

```
    //////获得Editor、获得Document、获得DocumentRoot
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
    CATDocument *pDoc = pEditor->GetDocument();
    CATIDocRoots* piDocRootsOnDoc = NULL;
    HRESULT rc = pDoc->QueryInterface(IID_CATIDocRoots,
                                (void**) &piDocRootsOnDoc);
    /////获得Product文件的Root Product，是第一个元素
    CATListValCATBaseUnknown_var* pRootProducts = piDocRootsOnDoc->GiveDocRoots();
    CATIProduct_var spRootProduct = NULL_var;
    if (pRootProducts && pRootProducts->Size())          //如果pRootProduct不为空，
```

```
获得第一个元素
    {
        ShowText("pRootProducts", pRootProducts->Size());
        spRootProduct = (*pRootProducts)[1];
        delete pRootProducts;
        pRootProducts = NULL;
    }
    piDocRootsOnDoc->Release();
    piDocRootsOnDoc = NULL;
    ////// Get CATIProduct handle on the root product.
    CATIProduct *piProductOnRoot = NULL;
    rc = spRootProduct->QueryInterface(IID_CATIProduct,
                                       (void**) &piProductOnRoot);

    /* -----------------------------------------*/
    /*    Retrieves children under the root   */
    /* -----------------------------------------*/
    /// 从Root Product获得所有的Direct Childrens数量
    int nbOfDirectChidren = piProductOnRoot -> GetChildrenCount() ;
    /////// 从Root Product开始获得所有的Children
    CATListValCATBaseUnknown_var*   ListChildren =
piProductOnRoot->GetAllChildren();
    CATListValCATBaseUnknown_var*   DirectListChildren =
piProductOnRoot->GetChildren();

    //GetChildren是获得所有的直接Children，而GetAllChildren是获取所有的children;
    piProductOnRoot -> Release();//指针在使用后立即释放，防止内存溢出。
    piProductOnRoot = NULL;
    if(NULL != ListChildren)
    {
        int numberOfChildren = ListChildren->Size();     //获得链表长度（所有
children的个数）
        ShowText("numberofAllChildren",numberOfChildren);       //用对话框显示链表
长度（注意：这个函数是自己定义的）
        /* ------------------------------------------------------------*/
        /*     For each child, get its partNumber, and InstanceName   */
        /* ------------------------------------------------------------*/
        CATIProduct_var spChild = NULL_var;
        _MultiList001 ->ClearLine() ;//清空MultiList
        AddinMultiList("======All Children List====");
        for (int i=1;i<=numberOfChildren;i++)
        {
            spChild = (*ListChildren)[i];
/** @anchor err_3 spChild not tested before use ( if !! ) */
```

```
            CATUnicodeString partNumber = spChild -> GetPartNumber();   //获得Part
的名称
            CATUnicodeString instanceName (" ");
            rc = spChild -> GetPrdInstanceName ( instanceName ) ;       //获得part
的实例化名称
            //将Part的实例化名称添加到MultiList//将Part的名称添加到MultiList（注意
这个函数是自己定义的）
            AddinMultiList(partNumber+"  ,  "+instanceName);
        }
        delete ListChildren;
        ListChildren=NULL;
    }
    if(NULL != DirectListChildren)
    {
        int numberOfChildren = DirectListChildren->Size();       //获得链表长度
（direct children的个数）
        ShowText("numberofDirectChildren",numberOfChildren);     //用对话框显示链表
长度（注意：这个函数是自己定义的）
        CATIProduct_var spChild = NULL_var;
        AddinMultiList("======Direct Children List====");
        for (int i=1;i<=numberOfChildren;i++)
        {
            spChild = (*DirectListChildren)[i];
/** @anchor err_3 spChild not tested before use ( if !! ) */
            CATUnicodeString partNumber = spChild -> GetPartNumber();   //获得Part
的名称
            CATUnicodeString instanceName (" ");
            rc = spChild -> GetPrdInstanceName ( instanceName ) ;       //获得part
的实例化名称
            //将Part的实例化名称添加到MultiList//将Part的名称添加到MultiList（注意
这个函数是自己定义的）
            AddinMultiList(partNumber+"  ,  "+instanceName);
        }
        delete DirectListChildren;
        DirectListChildren=NULL;
    }
```

6.2 控件中添加信息显示代码

以上代码包含了两个自己写的函数：AddinMultiList()和 ShowText；

在 BITCreateAssemDlg.h 文件里添加函数的声明代码：

```
private:
    void BITCreateAssemDlg::ShowText(CATUnicodeString ,int );
    //ShowText的声明，用以显示字符串和数字
```

```
    void BITCreateAssemDlg::AddinMultiList(CATUnicodeString );
//AddinMultiList的声明，用以将字符串添加在MultiList中
     CATDlgNotify *      _OpenNotify;
//声明全局变量_OpenNotify,用以打开消息框
```

由于在ShowText中要用到全局变量_OpenNotify，所以要有_OpenNotify的声明

在 BITCreateAssemDlg.cpp 文件里添加函数的实现代码：

```
void BITCreateAssemDlg::ShowText(CATUnicodeString istring,int Num)
{
    ////////////ShowText必须要有两个参数如果只显示数字则：第一个参数为"",如果只显示字符串，
第二个参数为0
    if( Num!=0)
    {
        CATUnicodeString istr;
        istr.BuildFromNum(Num,"%d");   //将数字转换为字符串函数BuildFromNum
        _OpenNotify = new CATDlgNotify(this, "", CATDlgNfyOK);
        _OpenNotify->SetText(istring+"\n"+istr);
        _OpenNotify->SetVisibility(CATDlgShow);
    }elseif( Num==0)
    {
        _OpenNotify = new CATDlgNotify(this, "", CATDlgNfyOK);
        _OpenNotify->SetText(istring);
        _OpenNotify->SetVisibility(CATDlgShow);
    }
}
void BITCreateAssemDlg::AddinMultiList(CATUnicodeString ColumnTitles)
{
    ////////////将字符创添加到MultiList001中，第一个参数表示为第一列，第二个参数是要添加
字符串的地址
        _MultiList001 ->SetColumnItems(0,&ColumnTitles,1,-1,CATDlgDataAdd);
}
```

添加头文件：

```
#include"CATFrmEditor.h"
#include"CATLib.h"
// ObjectModelerBase Framework
#include"CATDocumentServices.h"
#include"CATSessionServices.h"
#include"CATIDocRoots.h"
#include"CATDocument.h"
#include"CATInit.h"
#include"CATIPrtContainer.h"
#include"CATIPrtPart.h"
#include"CATIDescendants.h"
#include"CATIPartRequest.h"
#include"CATIBodyRequest.h"
```

```
#include"CATIGeometricalElement.h"
// ProductStructure Framework
#include"CATIProduct.h"
#include"CATBody.h"
#include"CATICGMObject.h"
#include"CATISchSession.h"
#include"CATIContainerOfDocument.h"
#include"SEQUENCE_octetColl.h"
```

这些头文件未必都会在上述程序中用到，也有可能添加不全，没关系，使用 CAAV5 Help Viewer 查找报错的声明，在最后可以看到：

This object is included in the file: **CATIProduct.h**
If needed, your Imakefile.mk should include the module: **CATProductStructure1**

只需要将这个.h 文件写入头文件即可。

6.3 运行获得结果

生成->mkmk

窗口->Open runtime window->cnext;

打开一个 Product 文件（），例如 ProductForTest.CATProduct

装配 a->装配->component1

（注意：component1 必须在 Product 环境下运行）

7. 遍历零件（Part） 树的实现

7.1 添加遍历代码

打开 BITCreateAssemDlg.cpp，找到

```
void BITCreateAssemDlg::OnSelectObjSndBtnPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{}
```

在大括号中添加：

```
////////用GetTree获得Part文件的结构树
int Check=GetTrees();//GetTree是自己写的函数，必须有声明、实现
if(Check==0)        //用返回值Check来判断在哪一步出错
{
    ShowText("成功",0);
}else
{
    ShowText("失败\n",Check);
}
```

7.2 获取结构树

以上代码用到了 GetTree，这是一个自己写的函数

在 BITCreateAssemDlg.h 文件中添加 GetTree 声明代码：

```
public:
int BITCreateAssemDlg::GetTrees();//GetTree声明
void BITCreateAssemDlg::PrintGeometricalFeaturesSetsResult(const
CATLISTV(CATBaseUnknown_var) &);
void BITCreateAssemDlg::PrintGeometricalFeaturesSetsResult2(const
CATLISTV(CATISpecObject_var) &) ;
```

同时声明了在 GetTree 中用到的两个显示几何特征函数：

PrintGeometricalFeaturesSetsResult 和 PrintGeometricalFeaturesSetsResult2

在 BITCreateAssemDlg.cpp 文件中添加 GetTree 实现：

```
int  BITCreateAssemDlg::GetTrees()
{
    HRESULT rc;
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();  //获得Editor
    CATDocument *pDoc = pEditor->GetDocument();              //获得Document
    if (  NULL ==pDoc  )
    {
```

```
        return 2;
    }
    //
    // 4- Queries on the document to get the root container
    //
    CATInit *pDocAsInit = NULL ;
    rc = pDoc->QueryInterface(IID_CATInit, (void**)&pDocAsInit);//从doc中获取prt文件的初始化
    if ( FAILED(rc) )
    {
        return 3 ;
    }
    CATIPrtContainer *pSpecContainer = NULL ; //获取prtContainer,采用都是获取接构container的方法2
    pSpecContainer = (CATIPrtContainer*)pDocAsInit->GetRootContainer("CATIPrtContainer");
    if ( NULL == pSpecContainer )
    {
        return 4 ;
    }
    pDocAsInit->Release();//编程好习惯，内存释放
    pDocAsInit = NULL ;
    //
    // 5- Retrieves on the root container (CATPrtCont) to get the Part feature（零件特征）
    //
    CATIPrtPart_var spPart = pSpecContainer->GetPart();//获取Part特征
    if ( NULL_var == spPart )
    {
        return 5 ;
    }
    pSpecContainer->Release();
    pSpecContainer = NULL ;
    //
    // 6- Retrieves the list of geometrical features set using CATIDescendants ...
    //
    CATIDescendants *pPartAsDescendants = 0;
    rc = spPart->QueryInterface(IID_CATIDescendants, (void**)&pPartAsDescendants) ;
    if ( FAILED(rc) )
    {
        return 6 ;
    }
    _MultiList001->ClearLine();//清空列表
    AddinMultiList("===============Part Structure=============");
```

```
    // 6-1 Extracts the lists of its Body Features
    CATLISTV(CATISpecObject_var) BodyListDesc;
    pPartAsDescendants->GetAllChildren("CATIMechanicalTool", BodyListDesc);//
    AddinMultiList("---------CATIMechanicalTool---------");
    PrintGeometricalFeaturesSetsResult2(BodyListDesc);
    // 6-2 Extracts the lists of its OGS
    CATLISTV(CATISpecObject_var) OGSList;
    pPartAsDescendants->GetAllChildren("CATIMmiOrderedGeometricalSet",OGSList);//
有序几何图形集
    AddinMultiList("-----CATIMmiOrderedGeometricalSet------");
    PrintGeometricalFeaturesSetsResult2(OGSList);
    // 6-3 Extracts the lists of its GS
    CATLISTV(CATISpecObject_var) GSList;
    pPartAsDescendants->GetAllChildren("CATIMmiNonOrderedGeometricalSet",GSList);
//几何图形集
    AddinMultiList("-----CATIMmiNonOrderedGeometricalSet------");
     PrintGeometricalFeaturesSetsResult2(GSList);

    pPartAsDescendants->Release();
    pPartAsDescendants = NULL ;
    //
    // 7- Retrieves the list of geometrical features set using CATIPartRequest ...
    //
    CATIPartRequest *pPartAsRequest = 0;
    rc = spPart->QueryInterface(IID_CATIPartRequest, (void**)&pPartAsRequest) ;
    if ( FAILED(rc) )
    {
       return 7 ;
    }
    constCATUnicodeString stdContext(" "); // Sets the context for topo bodies lookup
    // 7-1 Extracts the lists of its Body Features
    CATLISTV(CATBaseUnknown_var) BodyList;
    pPartAsRequest->GetSolidBodies(stdContext, BodyList);
    AddinMultiList("----------BodyList----------");
    PrintGeometricalFeaturesSetsResult(BodyList);
    // 7-2 Extracts the lists of its surfacic sets
    CATLISTV(CATBaseUnknown_var) SurfacicSetList;
    pPartAsRequest->GetSurfBodies(stdContext, SurfacicSetList);
    AddinMultiList("--------SurfacicSetList--------");
    PrintGeometricalFeaturesSetsResult(SurfacicSetList);
    pPartAsRequest->Release();
    pPartAsRequest = NULL ;
    return 0;
```

7.3 显示结构树的相关代码

```
void BITCreateAssemDlg::PrintGeometricalFeaturesSetsResult(const
CATLISTV(CATBaseUnknown_var) &iSet)
{
constCATUnicodeString stdContext("");

    ShowText("CATLISTV长度",iSet.Size());

//
// Processes all the sets
//
for(int curSetIdx=1; curSetIdx<=iSet.Size(); curSetIdx++)
  {
      CATBaseUnknown_var CurrentSet = iSet[curSetIdx] ;
if ( NULL_var == CurrentSet ) break ;

      CATIAlias_var aliasOnCurrentSet = CurrentSet ;
if ( NULL_var != aliasOnCurrentSet)
      {
        AddinMultiList(aliasOnCurrentSet->GetAlias());
          ShowText(aliasOnCurrentSet->GetAlias(),0) ;
      }
//
// 1- Retrieves the result
//
      CATLISTV(CATBaseUnknown_var) pListResult;

      CATIBodyRequest *pBodyRequestOnCurrentSet = NULL;
      HRESULT rc = CurrentSet->QueryInterface(IID_CATIBodyRequest,
(void**)&pBodyRequestOnCurrentSet);
if ( SUCCEEDED(rc) )
      {

         rc = pBodyRequestOnCurrentSet->GetResults(stdContext, pListResult);
if ( SUCCEEDED(rc) )
         {
int SizeListResult = pListResult.Size() ;
//
// 2- Processes each element of the list of results
//
for(int curFeatIdx=1; curFeatIdx<=SizeListResult; curFeatIdx++)
            {
               CATBaseUnknown_var CurrentFeat = pListResult[curFeatIdx] ;
```

```
if ( NULL_var == CurrentFeat ) break ;

                CATIAlias_var aliasOnCurFeat =  CurrentFeat ;
if ( NULL_var != aliasOnCurFeat)
                {
                   AddinMultiList(aliasOnCurFeat->GetAlias());
                }

//
// Gets the topological results linked to the feature
//
                CATIGeometricalElement *pGeomEltOnCurFeat = 0;
                rc = CurrentFeat->QueryInterface(IID_CATIGeometricalElement,

(void**)&pGeomEltOnCurFeat);
if( SUCCEEDED(rc) )
                {
                   CATBody_var ResultBody = pGeomEltOnCurFeat->GetBodyResult();

if( NULL_var != ResultBody )
                   {
                      CATICGMObject *pCurTopo = NULL ;

rc=ResultBody->QueryInterface(IID_CATICGMObject,(void**)&pCurTopo);
if( SUCCEEDED(rc) )
                      {
CATULONG32 curResultTag = pCurTopo->GetPersistentTag();


                         pCurTopo->Release();
                        pCurTopo = NULL ;
                      }else ;
                   }else ;

                   pGeomEltOnCurFeat->Release();
                   pGeomEltOnCurFeat = NULL ;
                }
             }
          }else;

          pBodyRequestOnCurrentSet->Release();
          pBodyRequestOnCurrentSet = NULL ;
       }
else
```

```
        {
         ;
        }
    }
}
void BITCreateAssemDlg::PrintGeometricalFeaturesSetsResult2(const
CATLISTV(CATISpecObject_var) &iSetSpecObj)
{
   CATLISTV(CATBaseUnknown_var) iSetBU ;
for (int i=1; i<=iSetSpecObj.Size(); i++)
   {
      iSetBU.Append(iSetSpecObj[i]);
   }
   PrintGeometricalFeaturesSetsResult(iSetBU);
}
```

7.4 运行

生成->mkmk

窗口->Open runtime window->cnext;

打开一个 Part 文件（），例如 PartCell.CATPart

装配 a->装配->component2

（注意：component2 必须在 CATPart 环境下运行）

8. 从装配体（Product）中获取零件（Part）

8.1 添加第三个按钮的响应函数代码

从 Product 文件获取每一个 Part 文件的代码主要由两部分，第一部分主要功能与“Product 结构”按钮功能类似，用以分析 Product 文件结构。第二部分代码功能与“Part 结构”按钮功能类似，主要用以分析 Part 文件。这两部分的衔接是通过对 Product 的每个 component 获取 ReferenceProduct，再通过 ReferenceProduct 获取 Part 文档，再从 Part 文档获取 container，从 container 获取结构 container，然后可以获取特征，输出到 multilist 框中。

按钮的回调为：

```
void BITCreateAssemDlg::OnCreateAssemBtnPushBActivateNotification(CATCommand* cmd,
CATNotification* evt, CATCommandClientData data)
{
int Check = GetTreesInProduct();
ShowText("",Check);
 }
```

8.2 获取装配体（Product）结构树

GetTreesInProduct()是自定义的获取 Product 结构的函数，它是自己定义的函数，必须在头文件中进行声明。声明代码如下所示：

```
public:    int BITCreateAssemDlg::GetTreesInProduct();
```

在这个函数的最后部分写了获取 referenceProduct，Part 文档，Container 和结构 container 的代码，具体实现为：

```
int  BITCreateAssemDlg::GetTreesInProduct()
{
    //////获得Editor、获得Document、获得DocumentRoot
    CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
    CATDocument *pDoc = pEditor->GetDocument();
    CATIDocRoots* piDocRootsOnDoc = NULL;
    HRESULT rc = pDoc->QueryInterface(IID_CATIDocRoots,
                                (void**) &piDocRootsOnDoc);
    /////获得Product的Root，是第一个元素
    CATListValCATBaseUnknown_var* pRootProducts = piDocRootsOnDoc->GiveDocRoots();
    CATIProduct_var spRootProduct = NULL_var;
    if (pRootProducts && pRootProducts->Size())          //如果pRootProduct不为空，
获得第一个元素
    {
        spRootProduct = (*pRootProducts)[1];
        delete pRootProducts;
        pRootProducts = NULL;
```

```
    }
    piDocRootsOnDoc->Release();
    piDocRootsOnDoc = NULL;
    ////// Get CATIProduct handle on the root product.
    CATIProduct *piProductOnRoot = NULL;
    rc = spRootProduct->QueryInterface(IID_CATIProduct,
                                       (void**) &piProductOnRoot);
    /// 从Root Product获得所有的Direct Childrens数量
    int nbOfDirectChidren = piProductOnRoot -> GetChildrenCount() ;
    /////// 从Root Product开始获得所有的Children
    CATListValCATBaseUnknown_var*   ListChildren =
piProductOnRoot->GetAllChildren();
    //GetChildren是获得所有的直接Children，而GetAllChildren是获取所有的children;
    piProductOnRoot -> Release();//指针在使用后立即释放，防止内存溢出。
    piProductOnRoot = NULL;
    if(NULL != ListChildren)
    {
        int numberOfChildren = ListChildren->Size();     //获得链表长度（所有
children的个数）
        CATIProduct_var spChildPrdt = NULL_var;
        _MultiList001 ->ClearLine() ;//清空MultiList
        AddinMultiList("======All Children List====");
        for (int i=1;i<=numberOfChildren;i++)
        {
            spChildPrdt = (*ListChildren)[i];
/** @anchor err_3 spChild not tested before use ( if !! ) */
            CATUnicodeString partNumber = spChildPrdt -> GetPartNumber();  //获得
Part的名称
            CATUnicodeString instanceName (" ");
            rc = spChildPrdt -> GetPrdInstanceName ( instanceName ) ;      //获得
part的实例化名称
            //将Part的实例化名称添加到MultiList//将Part的名称添加到MultiList（注意
这个函数是自己定义的）
            AddinMultiList(partNumber+"  ,  "+instanceName);
            CATIProduct_var ispProduct= spChildPrdt;
            CATIProduct_var spRef = ispProduct->GetReferenceProduct();//获取指向Part
文件的referenceProduct
            if ( NULL_var != spRef )
             {
                CATILinkableObject * piLinkableObject = NULL;
                rc = spRef->QueryInterface( IID_CATILinkableObject, (void**)&
piLinkableObject );
               if ( SUCCEEDED(rc) )
               {
```

```
                    CATDocument * pDocument = NULL ;
                    pDocument = piLinkableObject->GetDocument();//从Product的文档中
获得Part文档
                    CATIContainerOfDocument * pIContainerOfDocumentOnDoc = NULL ;//
获取Container
                     rc = pDocument->QueryInterface(IID_CATIContainerOfDocument,
(void**)&pIContainerOfDocumentOnDoc);
                    if (SUCCEEDED(rc) )
                    {
                       CATIContainer * pSpecContainer = NULL ;
                       rc =
pIContainerOfDocumentOnDoc->GetSpecContainer(pSpecContainer);//获取结构container，
采用方法一
                       if(SUCCEEDED(rc)) GetpartTreesinProduct(pSpecContainer);
                    }
                }
            }
        }
        delete ListChildren;
        ListChildren=NULL;
    }
return 0;
}
```

8.3 获取装配体（Product）中零件（Part）结构树

```
int  BITCreateAssemDlg::GetpartTreesinProduct(CATIContainer * pContainer)
{
    HRESULT rc;
    CATIPrtContainer *pSpecContainer = NULL;//这儿pContainer和pSpecContainer的内容
一致，只是换个接口表达。
    rc = pContainer->QueryInterface(IID_CATIPrtContainer,
(void**)&pSpecContainer) ;
    if ( NULL == pSpecContainer )
    {
       return 4 ;
    }
    // 5- Retrieves on the root container (CATPrtCont) to get the Part feature（零
件特征）
    CATIPrtPart_var spPart = pSpecContainer->GetPart();//获取PrtPart
    if ( NULL_var == spPart )
    {
       return 5 ;
    }
```

```
    pSpecContainer->Release();
    pSpecContainer = NULL ;
    // 6- Retrieves the list of geometrical features set using CATIDescendants ...
    CATIDescendants *pPartAsDescendants = 0;
    rc = spPart->QueryInterface(IID_CATIDescendants, (void**)&pPartAsDescendants) ;
    if ( FAILED(rc) )
    {
       return 6 ;
    }
    AddinMultiList("================Part Structure==============");
    // 6-1 Extracts the lists of its Body Features
    CATLISTV(CATISpecObject_var) BodyListDesc;
    pPartAsDescendants->GetAllChildren("CATIMechanicalTool", BodyListDesc);//
    AddinMultiList("---------CATIMechanicalTool---------");
    PrintGeometricalFeaturesSetsResult2(BodyListDesc);
    // 6-2 Extracts the lists of its OGS
    CATLISTV(CATISpecObject_var) OGSList;
    pPartAsDescendants->GetAllChildren("CATIMmiOrderedGeometricalSet",OGSList);//
有序几何图形集
    AddinMultiList("-----CATIMmiOrderedGeometricalSet------");
    PrintGeometricalFeaturesSetsResult2(OGSList);
    // 6-3 Extracts the lists of its GS
    CATLISTV(CATISpecObject_var) GSList;
    pPartAsDescendants->GetAllChildren("CATIMmiNonOrderedGeometricalSet",GSList);
//几何图形集
    AddinMultiList("-----CATIMmiNonOrderedGeometricalSet------");
     PrintGeometricalFeaturesSetsResult2(GSList);
    pPartAsDescendants->Release();
    pPartAsDescendants = NULL ;
    // 7- Retrieves the list of geometrical features set using CATIPartRequest ...
    CATIPartRequest *pPartAsRequest = 0;
    rc = spPart->QueryInterface(IID_CATIPartRequest, (void**)&pPartAsRequest) ;
    if ( FAILED(rc) )
    {
       return 7 ;
    }
    constCATUnicodeString stdContext(" "); // Sets the context for topo bodies lookup
    // 7-1 Extracts the lists of its Body Features
    CATLISTV(CATBaseUnknown_var) BodyList;
    pPartAsRequest->GetSolidBodies(stdContext, BodyList);
    AddinMultiList("----------BodyList----------");
    PrintGeometricalFeaturesSetsResult(BodyList);
    // 7-2 Extracts the lists of its surfacic sets
    CATLISTV(CATBaseUnknown_var) SurfacicSetList;
```

```
    pPartAsRequest->GetSurfBodies(stdContext, SurfacicSetList);
    AddinMultiList("--------SurfacicSetList--------");
    PrintGeometricalFeaturesSetsResult(SurfacicSetList);

    pPartAsRequest->Release();
    pPartAsRequest = NULL ;
    return 0;
}
```

8.4 运行

生成->mkmk

窗口->Open runtime window->cnext;

打开一个 Product 文件。

点击最后一个按钮

（注意：component2 必须在 CATProduct 环境下运行）