

第 10 讲 装配

目录

1	产品结构模型	2
2	给产品添加组件	2
2.1	添加组件的全局函数	3
2.1.1	创建一个新的 CATProduct 文件并初始化产品结构	3
2.1.2	添加已存在的 Part 文件	4
2.1.3	添加新的 Part 组件	4
2.1.4	添加新的 Product 组件	5
2.1.5	保存新的 Product 和 Part 文件	5
2.2	产品的 CATProduct 接口	5
2.2.1	添加已经存在的组件	5
2.2.2	常用的 CATProduct 方法	6
3	产品属性	8
3.1	获取一般属性	8
3.1.1	加载 Product 文档并初始化	8
3.1.2	获取所有产品	9
3.1.3	显示产品一般属性	10
3.2	获取全部属性	11
3.3	创建一个属性	11
4	产品的发布	12
4.1	加载并获取 Product 文档的产品	12
4.2	显示产品已经发布的内容	13
4.3	PrintPublications 的定义	13
4.4	修改产品的发布	15
4.5	保存修改的文档	16
5	装配约束	16
5.1	装配设计中的约束类型	16
5.1.1	固定约束	16
5.1.2	偏移约束	16
5.1.3	重合约束	17
5.1.4	角度约束	17
5.1.5	接触约束	17
5.1.6	约束类型数值说明	18
5.2	创建装配约束	18
5.2.1	创建固定约束	18
5.2.2	创建平行约束	19
6	代码实例	20

1 产品结构模型

产品是在工业范围内可以设计、制造和生产的東西。例如，一个产品可以是一个汽车，真空吸尘器或一个钻头。

一个产品是由不同的组件组成。当你设计一个产品，你必须确定被用来创建什么组件。然后，逐个设计每个元件。CAA 中的 **ProductStructure** 框架允许不同的组件组装成最终产品的一种组织和管理工具。此组件产生的是树形式的产品结构。

一个产品的组件本身可能是产品。这种设计允许重复使用的组成产品的组件，因为这样的话，一旦一个部件已经以产品的形式被创建，它可以被重新用于组成其它产品。

在 CATIA v5 中，存在一个专用于管理和存储与产品相关的数据结构的特定文档，它被称为一个 **CATProduct** 文档，它包含了具体的根产品（**root Product**）。产品结构的分支都开始于根产品对象。

所以，基本上，创造产品结构，您必须通过以下几步：

1. 创建一个 **CATProduct** 文档
2. 获取该文档的根产品元素
3. 给根产品添加足够多的新组件（零件或产品的形式）或已经定义的其他产品
4. 给添加的组件创建约束。

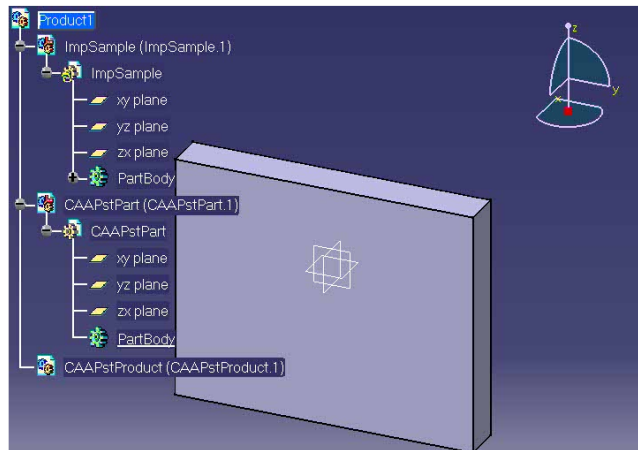
在 CATIAV5 版本中只有 **CATPart** 文件和 **CATProduct** 文件有 **root Product** 元素，这就意味着只有这两个文件可以用来装配。导入装配的 **component** 称为参考。一个 **part** 或 **Product** 在一个装配体结构中由两个名称来定义：

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 给产品添加组件

这一小节的目的是要给一个产品结构添加新的和现有的组件。创建一个新的 **CATProduct** 文件并添加了 3 种类型的组件到根产品：现有的一个 **Part** 零件，一个新的 **Part** 零件和新的组件。



上图中初始化的产品文档具有一个称为“Product1”的根产品，导入已经存在的名称为“ImpSample”的 Part 文件，并且实例化，它的实例化名称为“ImpSample.1”。也可以新添加一个 Part 文件，它的名称为“CAAPstPart”，实例化名称为“CAAPstPart.1”。新添加一个 Product 文件，实例化名称为“CAAPstProduct.1”。这些新的 CATPart 和 CATProduct 文件将会随着根产品文件的保存一起保存。

有 6 个步骤可以完成添加组件的操作。

常用的添加组件方法有两种：用全局函数添加和用 CATProduct 接口添加。

2.1 添加组件的全局函数

2.1.1 创建一个新的 CATProduct 文件并初始化产品结构

首先创建 CATProduct 文件，代码如下所示：

```
CATDocument *pDoc = NULL;
rc = CATDocumentServices::New("Product", pDoc);
if ( FAILED(rc) || (NULL==pDoc) ) return ;
```

其次获取根产品（root Product），代码如下：

```
CATIDocRoots *piDocRootsOnDoc = NULL;
rc = pDoc->QueryInterface(IID_CATIDocRoots, (void**) &piDocRootsOnDoc);
if ( FAILED(rc) ) return ;
CATListValCATBaseUnknown_var *pRootProducts= piDocRootsOnDoc->GiveDocRoots();
CATIPProduct_var spRootProduct = NULL_var;
if (pRootProducts)
if (pRootProducts->Size())
{
spRootProduct = (*pRootProducts)[1];
delete pRootProducts;
pRootProducts = NULL;
}
piDocRootsOnDoc->Release();
```

```

piDocRootsOnDoc=NULL;
// 获得了 root product 的 CATIPProduct 接口.
CATIPProduct *piProductOnRoot = NULL;
rc = spRootProduct->QueryInterface(IID_CATIPProduct, (void**) &piProductOnRoot);

```

为了初始化 CATProduct 文档内的产品结构，现在有必要访问根产品。使用 CATIDocRoots 的 GiveDocRoots 方法可以访问根产品，它返回所有根的列表，列表中第一个是我们所期待的根产品。从根产品，可以得到这将 CATIPProduct 接口，它用来对产品进行添加新组件操作。

2.1.2 添加已存在的 Part 文件

(1) 加载已存在的 CATPart 文件。

加载代码如下：

```

CATDocument *pDocPart = NULL;
CATIPProduct *piInstanceProd = NULL;

// 加载文件
rc = CATDocumentServices::Open("D:\\123.CATPart",pDocPart);
if ( FAILED(rc) || (NULL==pDocPart) ) return ;

```

为了添加已经存在的 Part 组件，Part 文件必须加载，采用 CATDocumentServices 的 Open 方法可以加载文件。

(2) 在根产品下导入 CATPart 文件。

导入代码如下：

```

rc = ::AddExternalComponent(piProductOnRoot, pDocPart,&piInstanceProd);
if ( FAILED(rc) ) return ;

```

现在 Part 文件已经加载了，为了将其作为组件导入到根产品，使用全局函数 AddExternalComponent。这个函数的 3 个参数是：

piProductOnRoot – 导入文件位置，这里是根产品。

pDocPart – 要导入的文件。

piInstanceProd – 导入文件的实例化 CATIPProduct 接口

2.1.3 添加新的 Part 组件

(1) 在根产品下导入一个新的 CATPart 文件。

导入代码如下：

```

CATIPProduct *piInstanceProd2 = NULL;
CATUnicodeStringpartName="New123Part";

// 创建一个新的 CATPart 文件并将其导入到装配体中
rc = ::AddNewExternalComponent(piProductOnRoot, "Part",partName, &piInstanceProd2);
if ( FAILED(rc) ) return ;

```

(2) 获取导入的新 CATPart 文档指针

代码如下所示：

```
CATDocument *pPartDoc2 = NULL;  
CATUnicodeStringpartDocName = partName + ".CATPart";  
rc = CATDocumentServices::GetDocumentInSession (partDocName, pPartDoc2);
```

新 CATPart 文件的 CATDocument 指针在后边保存文件的时候用到。

2.1.4 添加新的 Product 组件

(1) 在根产品下导入一个新的 CATProduct 文件。

代码如下：

```
CATIPProduct *piInstanceProd3 = NULL;  
CATUnicodeStringprdtName (NewPrdt);  
rc = ::AddNewExternalComponent(piProductOnRoot, "Product",prdtName, &piInstanceProd3);  
if ( FAILED(rc) ) return ;  
piProductOnRoot -> Release();  
piProductOnRoot = NULL;
```

添加新的 Product 组件与添加新的 Part 组件及其相似。采用的方法为全局函数 AddNewExternalComponent。

(2) 获取导入的新 CATProduct 文档指针

代码如下所示：

```
CATDocument *pProdDoc = NULL;  
CATUnicodeStringproductDocName = prdtName + ".CATProduct";  
rc = CATDocumentServices::GetDocumentInSession (productDocName , pProdDoc);  
if ( FAILED(rc) ) return ;
```

2.1.5 保存新的 Product 和 Part 文件

保存 CATProduct 文档

代码如下：

```
rc = CATDocumentServices::SaveAs(*pDoc, argv[6]);  
if ( FAILED(rc) ) return ;
```

保存 CATPart 文档的方法与 CATProduct 文档的一样。

2.2 产品的 CATIPProduct 接口

2.2.1 添加已经存在的组件

(1) 获取将要被导入文档的根产品

代码如下所示：

```
HRESULT rc = E_FAIL;
if ( NULL != iDocument)
{
    // 获取要导入文件的 RootProduct
    CATIDocRoots *piDocRootsOnDoc = NULL;
    rc = iDocument->QueryInterface(IID_CATIDocRoots, (void**) &piDocRootsOnDoc);
    if ( FAILED(rc) ) return;
    CATListValCATBaseUnknown_var *pRootProducts = piDocRootsOnDoc->GiveDocRoots();
    CATIPProduct_varspRootProduct = NULL_var;
    if ( NULL != pRootProducts)
    if (pRootProducts->Size())
    {
        // root product 是链表的第一个元素
        spRootProduct = (*pRootProducts)[1];
        deletepRootProducts;
        pRootProducts = NULL;
    }
}
```

iDocument 是指向 Product 或 Part 文件的指针；值得注意的是若 iDocument 不是指向 Product 或 Part 文件的指针那么无法获取 root product。

（2）添加组件到产品结构中

添加一个 CATPart 或 CATProduct 文档的代码如下所示：

```
spProduct = iThisProd->AddProduct (spRootProduct);
```

若要导入文档的 root Product 确实获得，那么 spRootProduct 智能指针有值，只需要执行 CATIPProduct 接口的 AddProduct 方法即可从 spRootProduct 参数创建一个要导入文档的实例化。iThisProd 是指导入的组件放置的位置。spProduct 是实例化的结果。

从实例化的结果中在查询 CATIPProduct 接口，那么查询的 CATIPProduct 接口是导入完成后在结构树上的组件。查询 CATIPProduct 接口代码如下所示：

```
CATIPProduct*oNewProduct
rc = spProduct->QueryInterface(IID_CATIPProduct, (void**) &*oNewProduct);
获得的 oNewProduct 指针可以进行访问产品名称，添加组件等操作。
```

（3）添加新的产品或零件

首先同样需要创建新的 Part 或 Product 文档；然后获取其根产品，然后添加到目标产品中；最后将新建的文档保存即可。

2.2.2 常用的 CATIPProduct 方法

常用的 CATIPProduct 方法解释：

AddConnector(CATILinkableObject_var&,CATIConnector_var&)

//添加链接，在添加约束之前首先要添加链接。

AddProduct(CATIPProduct_var&,CATIContainer_var&)

//创建或从已有的文档添加组件

AddProducts(CATListValCATBaseUnknown_var&,CATIContainer_var&,CATListValCATBaseUnknown_var*&)

//添加多个组件

//这两个函数用来给装配体添加零组件。添加更新后会看的零件出现在原点处

FindInstance(CATIPProduct_var&)

//从 CATIPProduct 中返回实例化对象

GetAllChildren(char*)

//获取 CATIPProduct 的所有子产品

GetChildren(char*)

//获取 CATIPProduct 的直接子产品

//GetAllChildren 和 GetChildren 两个函数是经常用到的变量结构树的函数，从结构树中找到你要的零件就靠它了

GetChildrenCount()

//获取子产品的数量，就是看节点上有几个子节点

GetFatherProduct()

//获取节点的父节点，或者父产品

GetPartNumber()

//获取参考产品的名称/零件号（PartNumber）

GetPrdInstanceName(CATUnicodeString&)

//获取实例名称

//CATIA 中特别注意三种名称：partnumber-零件号；instanceName -实例名；文件名

GetReferenceProduct()

//获取参考产品，基于此可以进一步获取参考产品的文档

RemoveProduct(CATIPProduct_var&)

//删除装配体中某个实例化的产品

SetPartNumber(CATUnicodeString&)

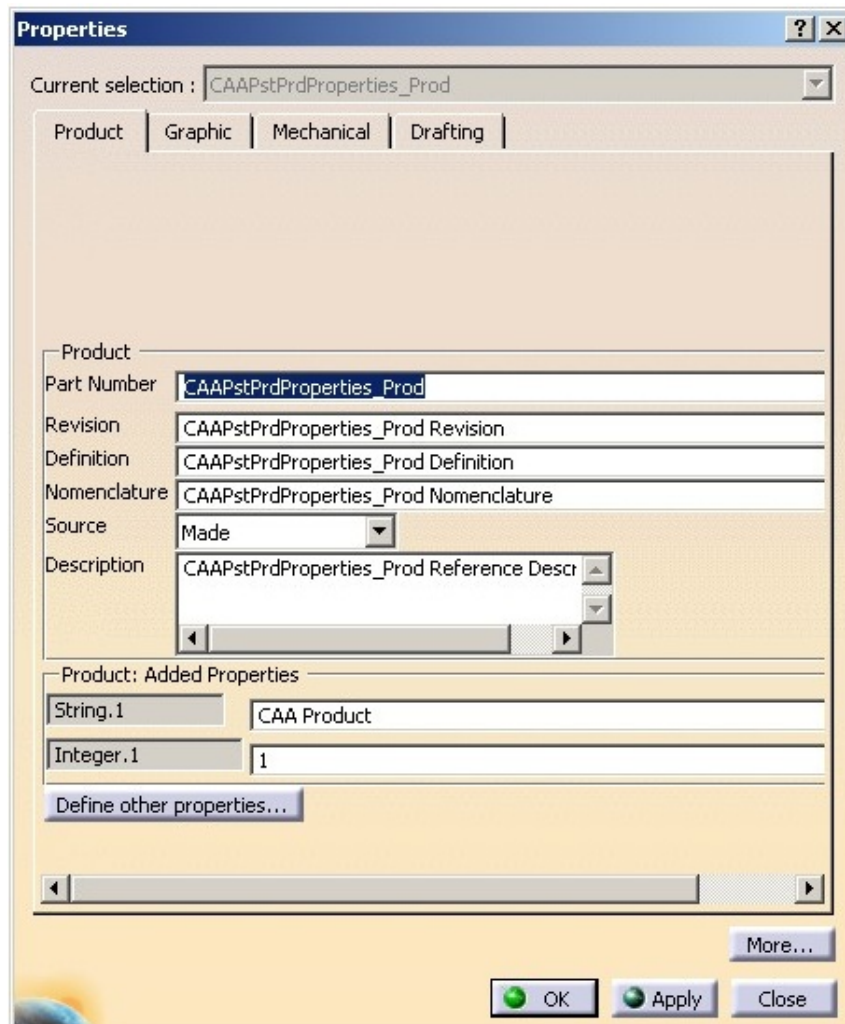
//设置参考产品的名称

SetPrdInstanceName(CATUnicodeString)

//设置产品实例化名称

3 产品属性

产品属性可以分成一般产品属性和用户定义属性。



3.1 获取一般属性

3.1.1 加载 Product 文档并初始化

加载产品文档可以通过 CATDocumentServices::Open 方法完成。加载之后获取文档的根产品（root Product），采用方法 CATIDocRoots::GiveDocRoots 可以获取文档的所有根，链表的第一个元素就是根产品。

代码如下所示：

```
CATDocument *pDoc = NULL;  
rc = CATDocumentServices::Open("D:\\123.CATProduct", pDoc); //加载文档  
CATIDocRoots* RootsOnDoc = NULL;
```



```

rc = pDoc->QueryInterface(IID_CATIDocRoots, (void**) &piDocRootsOnDoc);
CATListValCATBaseUnknown_var *pRootProducts= RootsOnDoc->GiveDocRoots();
CATIPProduct_var spRootProduct = NULL_var;
if (NULL != pRootProducts && pRootProducts->Size() > 0)
{
spRootProduct = (*pRootProducts)[1]; //获取文档根的第一个元素 root product
}

```

3.1.2 获取所有产品

(1) 显示根产品属性

获取了 root Product 后，需求再对其进行查询 CATIPProduct 接口，将获得的接口进行显示属性操作。专门定义了显示属性的函数：PrintPrdProperties；

查询 CATIPProduct 接口的代码为：

```

CATIPProduct *piProductOnRoot = NULL;
if (spRootProduct != NULL_var)
rc = spRootProduct->QueryInterface(IID_CATIPProduct, (void**) &piProductOnRoot);
//自己定义的显示根产品属性的函数
PrintPrdProperties(piProductOnRoot);

```

(2) 显示子产品（组件）属性

采用 CATIPProduct:: GetAllChildren 方法即可获得所有的子产品，对于每个子产品查询 CATIPProduct 接口，并将对应的 CATIPProduct 指针作为自定义函数 PrintPrdProperties 的输入参数，来显示子产品的属性。

```

CATListValCATBaseUnknown_var * childrenList = piProductOnRoot->GetAllChildren();
if (NULL == childrenList || childrenList->Size() <= 0) return;
CATIPProduct_var spChild = NULL_var;
int childrenCount = childrenList->Size();
for (int i = 1; i <= childrenCount; i++) //遍历子产品
{
spChild = (*childrenList)[i];
//获取子产品的 CATIPProduct 指针
CATIPProduct *piChildProduct = NULL;
if (NULL_var != spChild)
{
rc = spChild->QueryInterface(IID_CATIPProduct, (void**) &piChildProduct);
}
//显示子产品的属性
PrintPrdProperties(piChildProduct);
}

```

3.1.3 显示产品一般属性

产品属性可以通过 CATIPrdProperties 接口获取。PrintPrdProperties 函数首先显示了 Part Number 然后显示了实例化名称，接着用相应的方法获取每个属性，这些方法即为：GetNomenclature, GetRevision, GetSource, GetDefinition, GetDescriptionRef, GetDescriptionInst。

显示属性的代码为：

```
void PrintPrdProperties(CATIPrduct *iInstanceProd)
{
    HRESULT rc;
    CATUnicodeString oValue;
    if (NULL == iInstanceProd) return;
    oValue = iInstanceProd->GetPartNumber();
    rc = iInstanceProd->IsReference();
    if (FAILED(rc)) {
        rc = iInstanceProd->GetPrdInstanceName(oValue);
    }
    CATIPrdProperties _varspPrdProps(iInstanceProd);
    if (NULL != spPrdProps) {
        spPrdProps->GetNomenclature(oValue);
        spPrdProps->GetRevision(oValue);
        CatProductSource oSource;
        spPrdProps->GetSource(oSource);
        switch (oSource) {
        case catProductSourceUnknown:
            oValue = "Unknown";
            break;
        case catProductMade:
            oValue = "Made";
            break;
        case catProductBought:
            oValue = "Bought";
            break;
        default:
            oValue = "Undefined";
        }
        spPrdProps->GetDefinition(oValue);
        spPrdProps->GetDescriptionRef(oValue);
        spPrdProps->GetDescriptionInst(oValue);
    }
}
```

3.2 获取全部属性

获取全部属性的流程和获取一般属性类似，也要进行加载文档并初始化，获取包括根产品和子产品，最后显示产品属性，在之前显示一般属性时采用的是 CATIPrdProperties 接口，该接口不能显示所有接口，尤其对于自定义的属性无能为力。

采用 CATIAttributesDescriptor 和 CATIInstance 接口可以显示所有的属性。使用这个接口获取全部属性的代码为：

```
voidPrintAllProperties(CATIPProduct *iInstanceProd)
{
    HRESULT rc;
    if (iInstanceProd == NULL)        return;
    CATIAttributesDescription *piAttrDesc = NULL;
    rc = iInstanceProd->QueryInterface(IID_CATIAttributesDescription, (void **) &piAttrDesc);
    CATIInstance *piInstance = NULL;
    rc = iInstanceProd->QueryInterface(IID_CATIInstance, (void **) &piInstance);
    //获取所有的属性名称和值
    CATUnicodeStringpartNum = iInstanceProd->GetPartNumber();
    CATListValCATAttributeInfosattrInfoList;
    rc = piAttrDesc->List(&attrInfoList);
    intattrCount = attrInfoList.Size();
    for (int i = 1; i <= attrCount; i++)
    {
        CATAttributeInfosattrInfo = attrInfoList[i];
        constCATUnicodeString&propertyName = attrInfo.Name();
        constCATUnicodeString&valueType = attrInfo.Type()->Name();
        CATValue *pValue = piInstance->GetValue(propertyName);
        CATUnicodeStringvalueString;
        rc = pValue->AsString(valueString);
    }
}
```

3.3 创建一个属性

```
CATInit_varsplInitOnDoc = ipDoc;
if (NULL_var == splInitOnDoc)
{
    return ;
}
CATIContainer* pCont = (CATIContainer*) splInitOnDoc->GetRootContainer("CATIContainer");
CATICkeParmFactory_varspFact(pCont);
if(NULL_var == spFact)
{

```

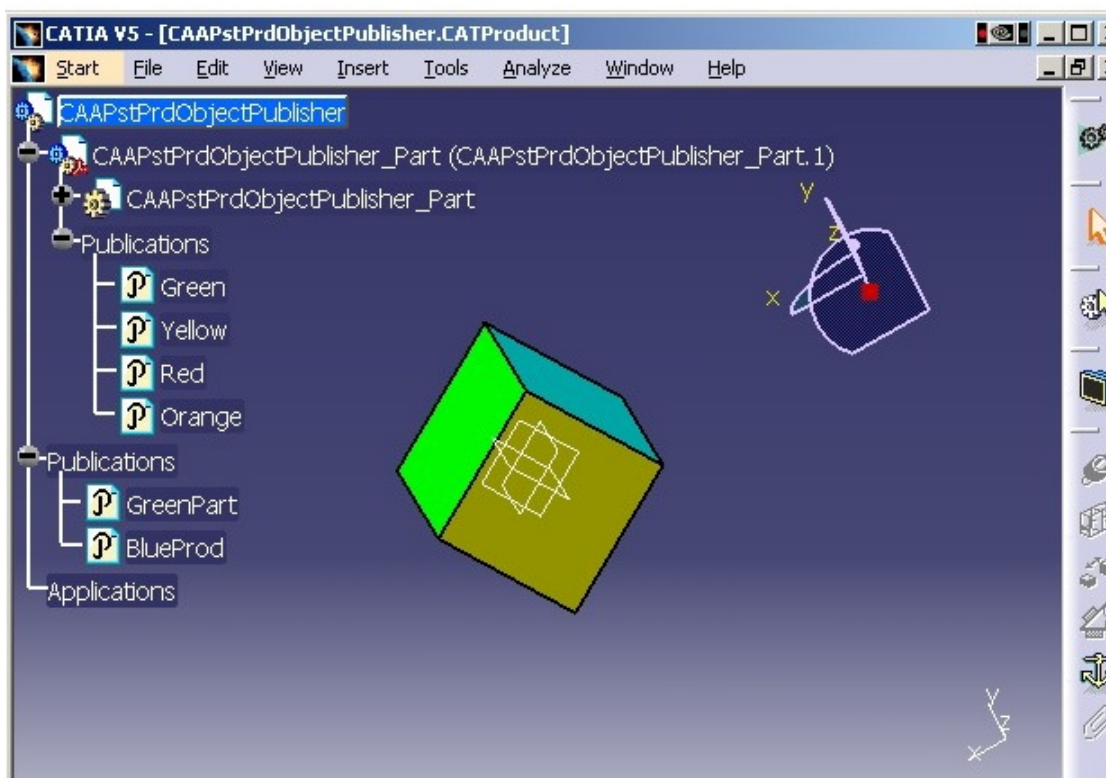
```

        return;
    }
    CATUnicodeString key = "Key";
    CATICkeParm_varspParm = spFact->CreateString(key, "Hallo");
    CATIPrdProperties_varspProp;
    spProp=ilInstanceProd->GetReferenceProduct();
    if(!spProp)
    {
        CATIParmPublisher* pPublisher = NULL;
        spProp->GetUserProperties(pPublisher, TRUE);
        pPublisher->Append(CATISpecObject_var(spParm));
    }
}

```

4 产品的发布

产品的发布主要包括显示根节点和它的子节点上的已有发布，添加新的发布，删除和修改已有发布。



接下来将一步步讲解实现过程。

4.1 加载并获取 Product 文档的产品

一般的，第一个步都是加载 Product 并初始化。然后获取 Product 文档的根产品（root Product）。要访问产品型文档，那么获取根产品是必不可少的一个步骤。采用 CATIDocRoots 接口的 GiveDocRoots 方法，其链表型结果的第一个值就是 root Product，然后在此根产品基础

上获取子产品。代码如下所示：

```
CATDocument *pDoc = NULL;
rc = CATDocumentServices::Open(inputDoc, pDoc);
CATIDocRoots* piDocRootsOnDoc = NULL;
rc = pDoc->QueryInterface(IID_CATIDocRoots, (void**) &piDocRootsOnDoc);
CATListValCATBaseUnknown_var* pRootProducts = piDocRootsOnDoc->GiveDocRoots();
CATIPProduct_var spRootProduct = NULL_var;
if (NULL != pRootProducts && pRootProducts->Size() > 0)
{
    spRootProduct = (*pRootProducts)[1];
}
```

获取根产品后查询 CATIPProduct 接口即可访问其子产品。

代码如下所示：

```
CATIPProduct *piProductOnRoot = NULL;
if (NULL_var != spRootProduct)
    rc = spRootProduct->QueryInterface(IID_CATIPProduct, (void**) &piProductOnRoot); //获取子产品
CATListValCATBaseUnknown_var *pChildren = NULL;
pChildren = piProductOnRoot->GetAllChildren();
int childCount = (pChildren == NULL ? 0 : pChildren->Size());
```

4.2 显示产品已经发布的内容

在获取了根产品和子产品的 CATIPProduct 接口后就可以显示所有的发布。代码如下所示：

```
rc = PrintPublications(piProductOnRoot);
if (0 != rc) return;
for (int i = 1; i <= childCount; i++)
{
    rc = PrintPublications((CATIPProduct_var) (*pChildren)[i]);
    if (0 != rc) return;
}
```

其中 PrintPublications 函数是自定义的显示发布的函数。

4.3 PrintPublications 的定义

PrintPublications 函数的用途在于显示产品的发布，使用 CATIPrdObjectPublisher 接口可以进行的工作有：获取所有发布、获取最后发布的对象、从一个发布得到所有的发布、获取发布的直接对象。

（1）获取所有发布

代码如下所示：

```
static int PrintPublications(CATIPProduct *ipiProduct)
{
```

```

intrc;
CATIPrdObjectPublisher *piPublisher = NULL;

if (NULL == ipiProduct) return 0;
rc = ipiProduct->QueryInterface(IID_CATIPrdObjectPublisher, (void **) &piPublisher);
CATUnicodeStringproductName = ipiProduct->GetPartNumber();
    // 获取当前产品的所有发布内容
CATListValCATUnicodeString *pPubList = NULL;
intpubCount = piPublisher->ListPublications(pPubList);
if (0 == pubCount || NULL == pPubList)
{
pubCount = 0;
}

```

产品所有发布的内容都存储在 piPublisher 指针中。pPubList 存储了所有发布的名称。

(2) 获取最后发布的对象

采用 GetFinalObject 方法可以获取最后发布的对象。

```

for (int i = 1; i <= pubCount; i++)
{
    // 获取发布的名称
CATUnicodeString&pubName = (*pPubList)[i];
    // 通过发布名称定位到最后发布对象
CATBaseUnknown *pPubObject = piPublisher->GetFinalObject(pubName);
    if (NULL == pPubObject)return 10;//不是最后的发布对象
}

```

(3) 从一个发布得到所有的发布

```

    // List publications pointing to other sub publications
CATListValCATBaseUnknown_var *pSubPubrList = NULL;
CATListValCATUnicodeString *pSubPubNameList = NULL;
rc = piPublisher->IsFinallyPublished(pPubObject, pSubPubrList, pSubPubNameList);
if (rc&& NULL != pSubPubrList&& NULL != pSubPubNameList&&
pSubPubrList->Size() > 0 &&pSubPubNameList->Size() > 0)
{
    CATUnicodeStringcurName;
for (int i = 1; i <= pSubPubNameList->Size(); i++)
curName = (*pSubPubNameList)[i];
}

```

(4) 获取发布的直接对象

```

    // 获取发布的直接对象
CATBaseUnknown *pSubPubr = NULL;
CATUnicodeStringsubPubName;
rc = piPublisher->GetDirectObject(pubName, pSubPubr, subPubName);

```

4.4 修改产品的发布

CATIPrdObjectPublisher 接口也提供修改发布的方法。自定义函数 ModifyPublication 集中表现了这些修改方法，ModifyPublication 需要三个输入参数：要修改产品的 CATIPProduct、要修改发布名称、修改后的发布名。ModifyPublication 的具体定义需要以下几步进行，代码如下：

(1) 获取 CATIPrdObjectPublisher 接口

```
static int ModifyPublications(CATIPProduct *ipiProduct,
                             const char *ipubToDelete, const char *isubPubToPublish)
{
    int rc;
    CATBaseUnknown *pSubPubr = NULL;
    CATUnicodeString subPubName;
    CATIPrdObjectPublisher *piPublisher = NULL;
    if (NULL == ipiProduct || NULL == ipubToDelete || NULL == isubPubToPublish)
        return 0;
    // 获取 CATIPProduct 的 CATIPrdObjectPublisher 接口
    rc = ipiProduct->QueryInterface(IID_CATIPrdObjectPublisher, (void **) &piPublisher);
```

(2) 确认要删除的发布时子发布

```
    rc = piPublisher->GetDirectObject(ipubToDelete, pSubPubr, subPubName);
    if (2 != rc || NULL == pSubPubr)    return 0;
```

(3) 获取子发布的 CATIPrdObjectPublisher 接口

```
    CATIPrdObjectPublisher *piSubPublisher = NULL;
    rc = pSubPubr->QueryInterface(IID_CATIPrdObjectPublisher, (void **) &piSubPublisher);
```

(4) 从产品层删除发布

```
    //对产品的发布进行 Unpublish 操作
    rc = piPublisher->Unpublish(ipubToDelete);
    if (2 != rc)    return 0;
```

(5) 从子发布层删除发布

```
    rc = piSubPublisher->Unvaluate(subPubName);
    if (2 != rc)    return 0;
    rc = piSubPublisher->Unpublish(subPubName);
    if (2 != rc)    return 0;
```

(6) 用给定的名称创建子发布

```
    // 在根产品层新建发布并给它赋值
    CATBaseUnknown *pPubObject = NULL;
    pPubObject = piSubPublisher->GetFinalObject(isubPubToPublish);
```

(7) 用新名称创建新发布

```
    CATUnicodeString newPubName(isubPubToPublish);
    newPubName += "Prod";
    rc = piPublisher->Publish(newPubName);
```

(8) 给新发布赋值

```
rc = piPublisher->Valuate(newPubName, pPubObject);
```

4.5 保存修改的文档

文档保存代码如下所示：

```
chardocDir[CATMaxPathSize];
chardocName[CATMaxPathSize];
CATSplitPath(inputDoc, docDir, docName);
CATUnicodeStringnewDoc = CATUnicodeString(outputDir) + "New" + docName;
rc = CATDocumentServices::SaveAs(*pDoc, newDoc);
```

5 装配约束

约束可以对装配中的一个组件相对于另一个组件的位置进行定位，只需要对一个或多个组件定义希望的约束类型，系统将会自动将组件放置在希望的位置上。

5.1 装配设计中的约束类型


















5.1.1 固定约束

固定一个组件意味着在产品更新时阻止这个组件相对它的父级运动。选择组件的一个几何元素及可以对该组件进行固定操作。

- 一是通过固定装配环境中的几何坐标进行固定，这就意味着设置了绝对位置。
 - 二是通过与其他组件位置进行固定，这就意味着设置了相对另一个组件的固定位置。
- 默认的情形是设置绝对位置。

5.1.2 偏移约束












在两个组件之间定义偏移约束时，应该定义具体的偏移方向。偏移约束的值始终与偏移约束一起显示。以下表格显示了能够定义偏移的不同元素。

	Point	Line	Plane	Planar Face
Point	 (1)	 (1)	 (1)	 (1)
Line	 (1)	 (1)	 (1)	 (1)
Plane	 (1)	 (1)	 (1)	 (1)
Planar Face	 (1)	 (1)	 (1)	 (1)

从上述图表中可以看出点、线、面、实体表面两两之间都能定义偏移约束关系。











5.1.3 重合约束

重合约束用来对齐元素，根据选择的不同元素，可以得到同轴、同心、共面等结果。以下表格显示了能够定义重合的不同元素。

 Point	Point	Line	Plane	Planar Face
Point	 (2)	 (2)	 (2)	 (2)
Line	 (2)	 (2)	 (2)	 (2)
Plane	 (2)	 (2)	 (2)	 (2)
Planar Face	 (2)	 (2)	 (2)	 (2)





5.1.4 角度约束

角度约束分为三大类：角度，平行，垂直。当选择一个角度约束时，必须定义角度约束的值，值得注意的是这个值不能超过 90 度。以下表格显示了能够定义角度约束的不同元素。

 Line	Line	Plane	Planar Face
Line	 (6, 8, 11)	 (6, 8, 11)	 (6, 8, 11)
Plane	 (6, 8, 11)	 (6, 8, 11)	 (6, 8, 11)
Planar Face	 (6, 8, 11)	 (6, 8, 11)	 (6, 8, 11)

5.1.5 接触约束

接触约束可以在两个表面或平面之间创建，两个面之间的接触部分可以是一个平面（平面接触），一条直线（线接触），一个点（点接触）。以下表格显示了能够定义角度约束的不同元素。

 Planar Face	Planar Face	Sphere	Cylinder	Cone	Circle
Planar Face	 (20)	 (22)	 (21)		
Sphere	 (22)	 (20)		 (25)	 (25)
Cylinder	 (21)		 (20, 21)		
Cone		 (25)		 (20, 21)	 (25)
Circle		 (25)		 (25)	

5.1.6 约束类型数值说明

在以上图表中每个单元下都有个括弧数字，它的意思是这个约束元素对应的枚举类型的值。装配约束类型在 CATIA v5 中用 `CatConstraintType` 枚举出来。`CatConstraintType` 不仅仅枚举了装配约束还有其他零件草图尺寸约束等内容。装配约束枚举如下：

`catCstTypeReference`, 固定约束, 值为 0
`catCstTypeDistance`, 偏移约束, 值为 1
`catCstTypeOn`, 重合约束, 值为 2
`catCstTypeAngle`, 角度约束, 值为 6
`catCstTypePlanarAngle`, 平面角度约束, 值为 7
`catCstTypeParallelism`, 平行角度约束, 值为 8
`catCstTypePerpendicularity`, 垂直角度约束, 值为 11
`catCstTypeSurfContact`, 面接触约束, 值为 20
`catCstTypeLinContact`, 直线接触约束, 值为 21
`catCstTypePoncContact`, 点接触约束, 值为 22
`catCstTypeAnnulContact`, 曲线接触约束, 值为 25

5.2 创建装配约束

5.2.1 创建固定约束

(1) 加载产品文档

加载产品文档在上文已经介绍过，此处不再重复。

(2) 创建连接

采用 `GetProductConnector` 全局函数创建组件实例的连接。代码如下所示：

```
::GetProductConnector(pGeometry, pInstanceComponent, pActiveComponent  
, 0, pConnector, iCreation);
```

其中 `pGeometry` 输入参数是要引用的几何；

`pInstanceComponent` 输入参数是上述引用几何所在的组件；

`pActiveComponent` 输入参数是激活的组件，约束会创建在激活的组件中；

0 输入参数是搜索模式，0 意味着如果没有找到就创建连接；

`pConnector` 输出参数，创建好的链接；

`iCreation` 输出参数，创建模型；0 表示链接已存在并且已找到，1 表示链接已创建。

(3) 创建约束

采用 CreateConstraint 全局函数创建组件约束。代码如下所示：

```
rc = ::CreateConstraint(catCstTypeReference, ConnectorList, NULL, pActiveComponent, pCst);
```

其中 catCstTypeReference 输入参数表示要创建的约束类型为固定约束；

ConnectorList 输入参数，包含了在这个约束中用到的所有链接；

NULL 输入参数。约束值，对于角度约束和偏移约束特有的。固定约束无值；

pActiveComponent 输入参数，激活的组件；约束创建的位置；

pCst 输出参数，创建的约束特征。

5.2.2 创建平行约束

与创建固定约束类似。

（1）加载产品文档

加载产品文档在上文已经介绍过，此处不再重复。

（2）创建连接

采用 GetProductConnector 全局函数创建组件实例的连接。代码如下所示：

```
::GetProductConnector(pGeometry, pInstanceComponent, pActiveComponent, 0, pConnector, iCreation);
```

其中 pGeometry 输入参数是要引用的几何；

pInstanceComponent 输入参数是上述引用几何所在的组件；

pActiveComponent 输入参数是激活的组件，约束会创建在激活的组件中；

0 输入参数是搜索模式，0 意味着如果没有找到就创建连接；

pConnector 输出参数，创建好的链接；

iCreation 输出参数，创建模型；0 表示链接已存在并且已找到，1 表示链接已创建。

（3）创建约束

采用 CreateConstraint 全局函数创建组件约束。代码如下所示：

```
rc = ::CreateConstraint(catCstTypeParallelism, ConnectorList, NULL, pActiveComponent, pCst);
```

其中 catCstTypeReference 输入参数表示要创建的约束类型为固定约束；

ConnectorList 输入参数，包含了在这个约束中用到的所有链接；

NULL 输入参数。约束值，对于角度约束和偏移约束特有的。固定约束无值；

pActiveComponent 输入参数，激活的组件；约束创建的位置；

pCst 输出参数，创建的约束特征。

6 代码实例

来自实例train8code，其中SelectToConstraint.cpp文件的212行之后部分。

```
//////////1, 获取rootproduct
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
CATDocument *pDoc = pEditor->GetDocument();

CATIDocRoots *piDocRootsOnDoc = NULL;
rc = pDoc->QueryInterface(IID_CATIDocRoots, (void**) &piDocRootsOnDoc);
if ( FAILED(rc) ) return ;
CATListValCATBaseUnknown_var *pRootProducts =
piDocRootsOnDoc->GiveDocRoots();
CATIPrduct_var spRootProduct = NULL_var;//获取rootProduct
if (pRootProducts)
if (pRootProducts->Size())
{
    spRootProduct = (*pRootProducts)[1];
    delete pRootProducts;
    pRootProducts = NULL;
}
piDocRootsOnDoc->Release();
piDocRootsOnDoc=NULL;

//2, 获得装配元素的Connector
CATIConnector * pConnector1 = NULL;
CATILinkableObject_var spLinkObj1;

rc = m_spObjFst->QueryInterface(IID_CATILinkableObject , (void **) &spLinkObj1);
if ( SUCCEEDED(rc) )
{
int iCreation = 0;
rc = ::GetProductConnector(spLinkObj1,
                           m_pInstPrdt1,
                           (CATIPrduct *)spRootProduct,
                           0,
                           pConnector1,
                           iCreation);
}

//2, 获得装配元素的Connector
CATIConnector * pConnector2 = NULL;
CATILinkableObject_var spLinkObj2;
```

```

    rc = m_spObjFst->QueryInterface(IID_CATILinkableObject , (void **) &spLinkObj2);
if ( SUCCEEDED(rc) )
{
int iCreation = 0;
    rc = ::GetProductConnector(spLinkObj2,
                                m_pInstPrdt2,
                                (CATIPProduct *)spRootProduct,
                                0,
                                pConnector2,
                                iCreation);
}

```

//3, 创建约束

```

CATLISTV (CATBaseUnknown_var) ConnectorList;
ConnectorList.Append(pConnector1);
ConnectorList.Append(pConnector2);

CATICst *piCst = NULL;
rc = ::CreateConstraint(catCstType0n,
                        ConnectorList,
                        NULL,
                        (CATIPProduct *)spRootProduct,
                        &piCst);
CATIAConstraint* pCATIACst = NULL;
pCst->QueryInterface(IID_CATIAConstraint, (void**)&pCATIACst);

```