

第 6 讲知识工程

目录

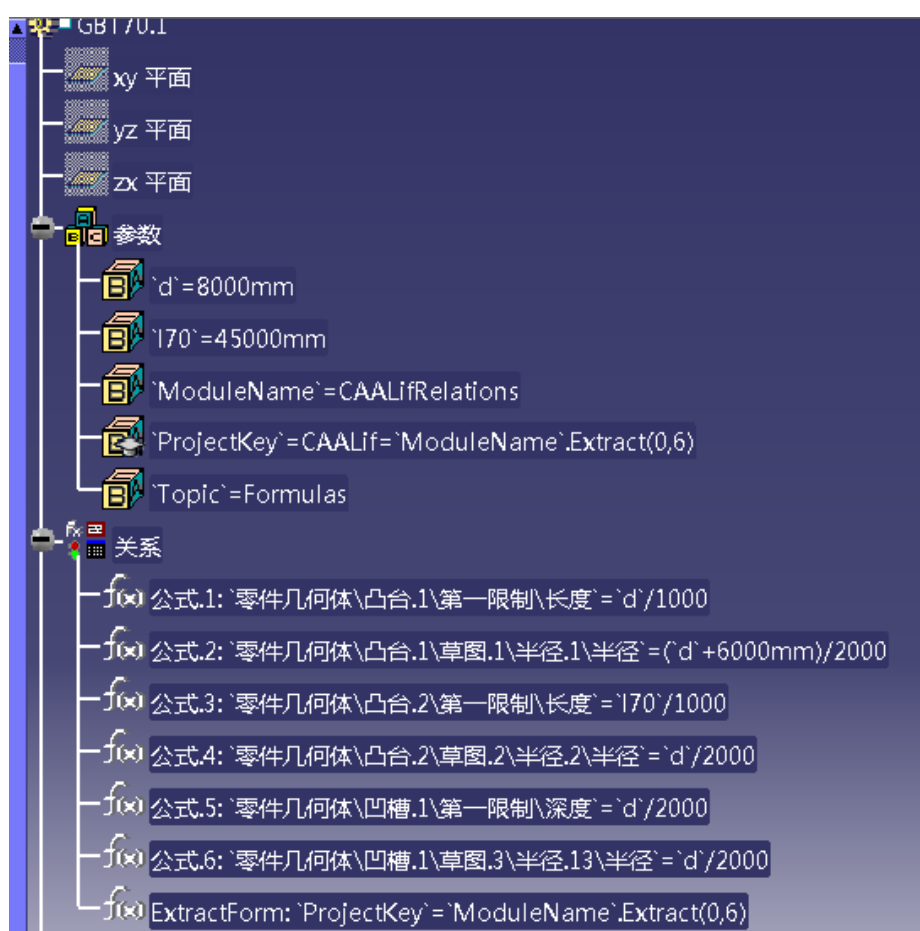
1	目标	错误！未定义书签。
1.1	创建参数和公式	3
1.2	读取参数并赋值	3
1.3	使用设计表	4
1.4	用户自定义特征	5
1.5	参考资料	5
2	知识工程综述	5
3	公式和参数	6
3.1	创建参数和公式	6
3.1.1	初始化	6
3.1.2	创建参数	7
3.1.3	参数发布	7
3.1.4	创建参数列表	8
3.1.5	创建公式	8
3.1.6	修改公式表达式	9
3.1.7	公式发布	9
3.1.8	激活公式	9
3.2	读取参数并赋值	9
3.2.1	读取参数初始化	9
3.2.2	从 Part 特征获取参数集列表	10
3.2.3	遍历参数集列表	10
3.2.4	读取参数集中的参数	10
3.2.5	修改参数	11
4	使用设计表	11
4.1	创建和使用设计表	11
4.1.1	初始化	11
4.1.2	创建参数	12
4.1.3	创建设计表	12
4.1.4	设计表和参数关联	12
4.1.5	获取链接数量	13
4.1.6	获取及设置当前配置	13
4.1.7	删除参数与设计表的联系	14
4.2	修改设计表的值	14
5	用户自定义特征	14
5.1	定义	14
5.2	创建用户自定义特征	16
5.2.1	获取用户自定义特征工厂	16
5.2.2	创建新特征参考	16
5.2.3	添加组件	17

5.2.4	检查组件	17
5.2.5	组件赋值	17
5.2.6	添加新特征到结构树	18
5.3	用户自定义特征实例化	18
5.3.1	获取要实例化的用户自定义特征	19
5.3.2	设置目标位置	19
5.3.3	给输入赋值	20
5.3.4	给发布参数赋值	20
5.3.5	实例化	21
5.3.6	修改新用户特征的名称	21
5.3.7	结束实例化过程	21
5.4	编辑用户自定义特征	22
5.4.1	获取用户特征	22
5.4.2	开始修改	23
5.4.3	修改输入	23
5.4.4	修改发布的参数	23
5.4.5	修改结束	24
6	知识工程功能实现	24
6.1	创建参数和公式	24
6.2	读取参数并赋值	26
6.3	使用设计表	27
7	额外的工程实例 P3	29

1 概述

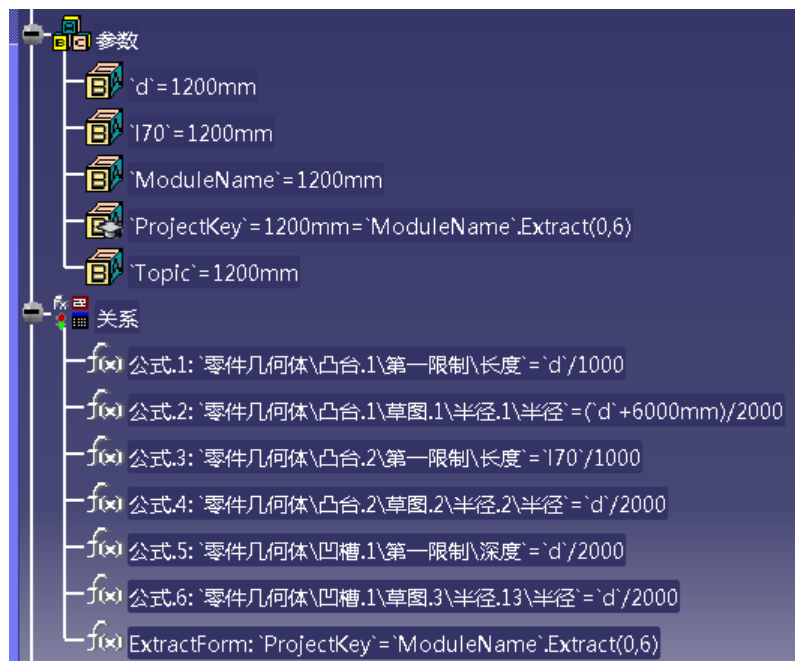
1.1 创建参数和公式

创建 3 个字符串形式的参数。创建一个公式。



1.2 读取参数并赋值

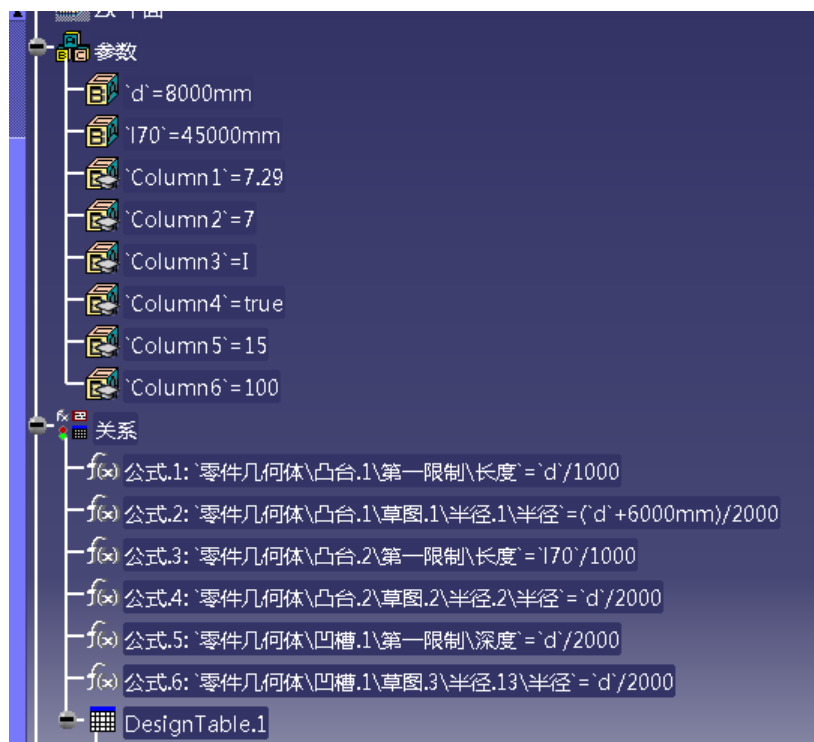
读取每个参数，并给每个参数赋值"1200mm"



1.3 使用设计表

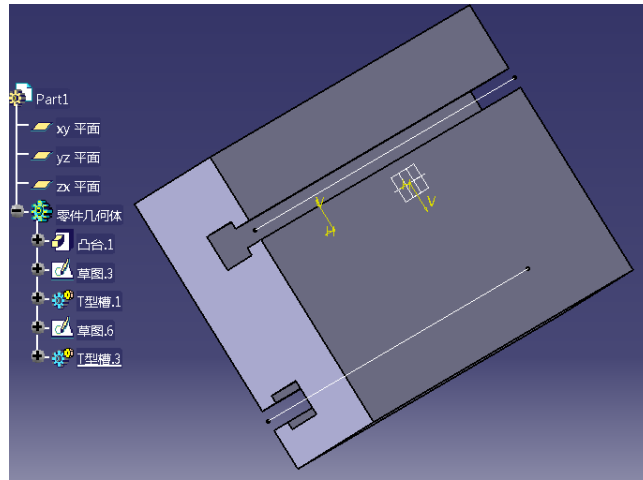
将设计表与参数关联。

DesignTable - 记事本					
文件(F)	编辑(E)	格式(O)	查看(V)	帮助(H)	
Column1	Column2	Column3	Column4	Column5	Column6
7	7.5	Test	False	15mm	10
7.29	7	I	True	15	100



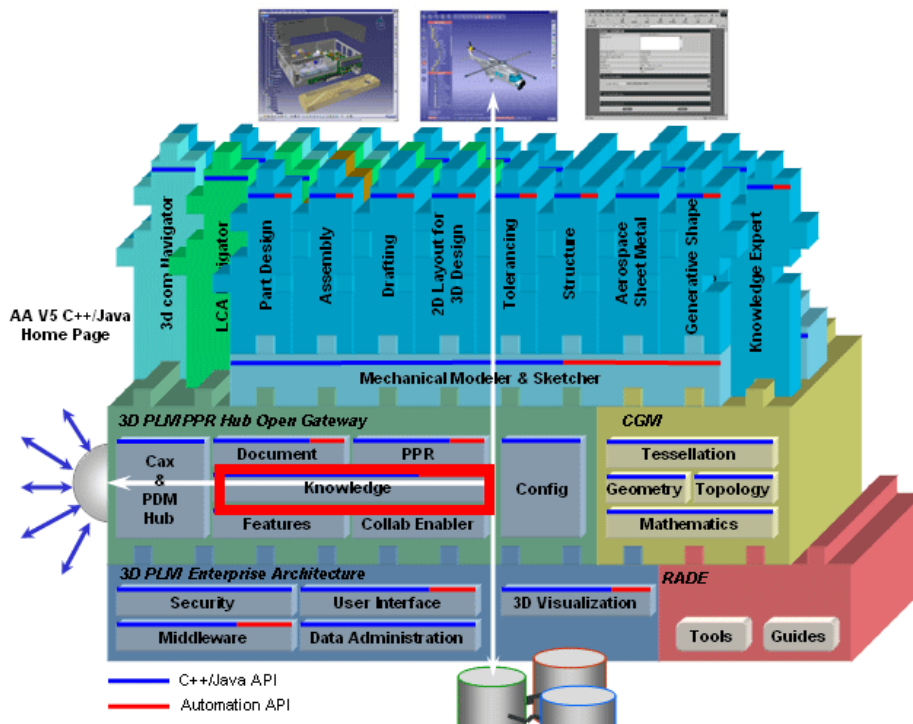
1.4 用户自定义特征

实例化 T 型槽用户特征



1.5 参考资料

在 Knowledge 模块中主要涉及参数、设计表、关系、公式。




在 MechanicalModeler&Sketcher 中主要为用户特征、模板建模方法。

2 知识工程综述

本章的目的是从开发者的角度解释使用参数和关系的好处。参数是当对对象进行操作时显

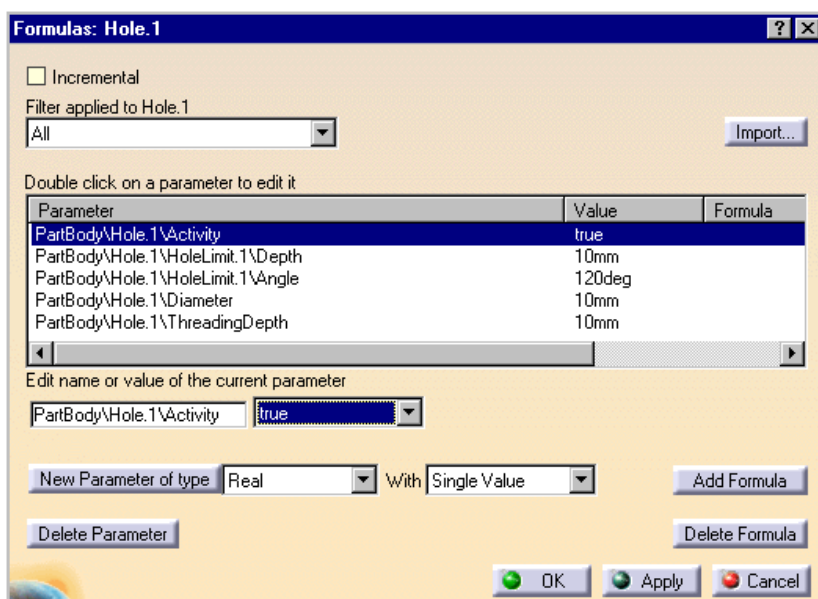
示或者修改的属性。关系可以定义一个参数与其他对象的联系。对于一个用户而言可以从公式对话框中创建参数。参数一般可以以文字形式表示。参数和关系都有知识工程接口框架进行管理。

以机械特征孔为例来说明知识工程接口工作模式，孔被创建的时候进行了类型定义（简单孔、阶梯孔等），伸展模式定义（直到最后，盲孔等），当然还有它的直径，深度，螺纹和其他内容。当对一个孔进行编辑操作时，可以从一个面板上看到所有上述内容，这些内容均由 CATIA 内部以特征属性的方式编码（CATISpecAttribute）。

当点击  按钮时，大多数孔的特征属性显示在参数列表中，当满足以下两个主要条件后均可用公式对话框编辑操作：

1.孔特征必须激活，是 CATIParmPublisher 的实现；2.特征属性必须属于是知识工程接口框架可以操作的范围。

实际上，所有的特征在被完成建模时就已经完成了特征激活。



从 $F(x)$ 可以进行孔特征大部分参数的修改，也可以给这些特征参数添加公式，从而用公式驱动产生参数值。

用户特征也是知识工程的重要内容，采用用户特征可以简化建模过程，提高设计效率。

3 公式和参数

3.1 创建参数和公式

3.1.1 初始化

在进行下边内容之前初始化环境。

初始化主要是指创建一个文件，并获取它的根 Container，并且给 CATICkeParmFactory 接口赋值。关于 CATICkeParmFactory 接口必须注意两点，一是它可以采用任何根 container 实现，二是它可以提供任何参数和关系的创建方法。初始化代码如下所示：

```
HRESULT rc;
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor(); //获得 Editor
CATDocument *pDoc = pEditor->GetDocument(); //获得 Document
if ( NULL ==pDoc ) return ;
CATInit *pInitOnDoc = NULL ;
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc); //从 doc 中获取 prt 文件的初始化
if ( FAILED(rc) ) return ;
pInitOnDoc->Init(TRUE);
CATIContainerOfDocument_var spNewDocContainer = pDoc;
CATIContainer *cont = NULL;
rc = spNewDocContainer->GetSpecContainer(cont); //获取结构 Container
if (FAILED(rc) || cont == NULL) return ;
CATICkeParmFactory_var spFact = cont;
```

3.1.2 创建参数

此处采用 CATICkeParmFactory 的方法创建 3 个字符串类型的参数。在一般的应用程序中这些参数已经创建，只需要将参数获得并且存放到一个链表中即可。创建参数代码如下所示：

```
CATICkeParm_var spString1=spFact->CreateString("ModuleName","CAALifRelations");
CATICkeParm_var spString2=spFact->CreateString("ProjectKey","");
CATICkeParm_var spString3=spFact->CreateString("Topic","Formulas");
```

使用 CATParmDictionary::VisibleParms 可以获取已经发布的特征参数列表。

注意创建不同类型的参数时使用的方法不同。

3.1.3 参数发布

创建的参数并不会公式在界面中显示出来，只有经过发布的参数才会显示给用户。发布参数主要用到 CATIParmPublisher 接口。发布的对象很多，可以发布参数，公式，甚至是几何特征，所以发布时不同的对象发布的方法略有不同。首先获取 Part 特征，以前讲过 Part 特征是最高级的特征，参数等要发布的对象就包含在 Part 特征中。从 Part 特征中查询 CATIParmPublisher 接口，再采用 CATICkeFunctionFactory::GetCurrentSet 方法获取当前发布的对象集，最后将要发布的对象链接到发布对象集。发布参数代码如下：

```
CATIPrtContainer *pIPrtCont = NULL; //结构 Container
rc = cont->QueryInterface(IID_CATIPrtContainer, (void**)&pIPrtCont); //
if(pIPrtCont==NULL) return;
CATISpecObject_var spSpecObjectOnPart = pIPrtCont->GetPart(); //获取 Part 特征
CATIPrtPart_var _spPart=NULL_var;
CATIPrtPart * pIPrtPart = NULL ;
```

```

spSpecObjectOnPart->QueryInterface(IID_CATIPrtPart,(void**) & _spPart ) ;//获取 Part 特征
if(_spPart==NULL_var) return;
CATIParmPublisher * opiPublisher;//从 Part 特征获取发布集
rc = _spPart -> QueryInterface(IID_CATIParmPublisher, (void**)&opiPublisher);
CATISpecObject_var spParameterSet = CATCkeGlobalFunctions::GetFunctionFactory()
    ->GetCurrentSet(CATICkeFunctionFactory::Parameter,opiPublisher,CATCke::True);// 设置发布集对
象

if(opiPublisher==NULL)return;
CATIParmPublisher_var spParmPublisher = spParameterSet;
if (!!spParmPublisher)//将要发布的内容链接到发布集
{
    spParmPublisher->Append(spString1);
    spParmPublisher->Append(spString2);
    spParmPublisher->Append(spString3);
}

```

3.1.4 创建参数列表

在创建关系或公式之前必须创建至少一个参数列表，所有公式中使用的参数都在参数列表中存储。公式约束参数，并且公式表达式中用到的参数必须在参数列表中。

基于上述内容，可以创建一个参数比公式需要的参数多的参数列表。

参数链表中的参数顺序可以以任何顺序存放，但是我们必须记住这个顺序，因为在创建公式的时候会使用这一顺序。

```

CATCkeListOf(Parm) pList;
pList.Append (spString1); // a1 = "ModuleName"
pList.Append (spString2); // a2 = "ProjectKey"
pList.Append (spString3); // a3 = "Topic"

```

如果修改参数列表，那么必须注意依赖于这个参数列表的公式和关系会改变。

3.1.5 创建公式

采用 CATICkeParmFactory:: CreateFormula 方法可以创建公式。在这儿创建一个公式，ProjectKey 值等于 ModuleName 的前 6 个字母。

在创建这个公式中需要使用 Extract 方法，它是知识工程中的一个关键词。用来返回给定位置之间的子字符串，Extract 的第一个参数是字符串首位置，第二个字符串是结束位置。、

这个公式它在结构树中具体表示为 “a1.Extract(0,6)”。创建公式的代码为：

```

CATICkeRelation_var
    spFormula1 = spFact->CreateFormula ("ExtractForm","", "",
    spString2, &list, "a1.Extract(0,6)", NULL_var, CATCke::False);
CreateFormula 方法的参数具体为：

```


参数 1 是公式的名称 ExtractForm;参数 2 应该是空字符串;参数 3 应该是空字符串;参数 4 是要约束的变量,这儿是 ProjectKey;参数 5 是在这个公式中要用到的所有变量列表,此处是 list;参数 6 是公式的约束变量的表达式,此处为“a1.Extract(0,6)”;值得注意的是 a1 与变量列表中的第一个元素 ModuleName 对应。参数 7 应该是空智能指针 NULL_var;参数 8 应该是 CATCke::False;这儿采用了 a1 这种形式,所以应该为 False。

3.1.6 修改公式表达式

采用 CATICkeRelationExp::Modify 方法可以修改公式表达式,在修改的时候表达式处于未激活状态,如果在再次激活之前显示变量的值,那么显示的是以前值而不是新修改的值。修改公式代码如下所示:

```
CATICkeRelationExp_var spExpform = spFormula1 ;  
spExpform->Modify(&list,"a3.Extract(0,5)", NULL_var, CATCke::False);
```

其中参数 1 是要用到的变量列表;参数 2 是公式的新表达式;参数 3 应该是空智能指针 NULL_var;参数 4 应该是 CATCke::False;

3.1.7 公式发布

发布公式与发布参数方法类似,唯一区别在于设置发布对象集的时候为 Relation;

```
CATISpecObject_var spParameterSet = CATCkeGlobalFunctions::GetFunctionFactory()  
->GetCurrentSet(CATICkeFunctionFactory::Relation,opiPublisher,CATCke::True);  
//设置发布集对象为 Relation
```

3.1.8 激活公式

采用 CATICkeRelation::Activate() 方法可以激活刚刚修改的公式。

```
spFormula1->Activate();
```

3.2 读取参数并赋值

在一个 Part 文件中可以由很多参数,这些参数组成不同的参数集,读取参数时需要遍历参数集,逐个读取参数。

3.2.1 读取参数初始化

代码如下所示:

```
HRESULT rc;  
CATBaseUnknown_var spBaseUknParm[20]={NULL};  
CATICkeParm* pCkeParam[20]={NULL};  
CATIPrtPart_var spPrtPart;//零件 part  
CATIContainerOfDocument_var spNewDocContainer = pPartDoc;
```

```

CATIContainer *cont = NULL;
rc = spNewDocContainer->GetSpecContainer(cont); //获取结构 container
if (FAILED(rc) || cont == NULL) return;
CATIPrtContainer *pIPrtCont = NULL; //结构 Container
rc = cont->QueryInterface(IID_CATIPrtContainer, (void**)&pIPrtCont); //
if(spPrtPart==NULL_var) return;

```

3.2.2 从 Part 特征获取参数集列表

代码如下：

```

if(pIPrtCont==NULL) return;
CATISpecObject_var spSpecObjectOnPart = pIPrtCont->GetPart(); //获取 Part 特征
spSpecObjectOnPart->QueryInterface(IID_CATIPrtPart,(void**) & spPrtPart ); //获取 Part 特征
CATIDescendants *pPartAsDescendants = 0; //从 Part 特征获取继承对象
rc = spPrtPart->QueryInterface(IID_CATIDescendants, (void**)&pPartAsDescendants) ;
//获得参数集列表
CATLISTV(CATISpecObject_var) ParmListDesc;
pPartAsDescendants->GetAllChildren("CATICkeParameterSet", ParmListDesc); //找到参数集对象
pPartAsDescendants->Release();
pPartAsDescendants = NULL ;

```

3.2.3 遍历参数集列表

代码如下：

```

//获得每一个参数集中的参数
for(int curSetIdx=1; curSetIdx<=ParmListDesc.Size(); curSetIdx++)
{
    CATICkeParameterSet_var CurrentSet = ParmListDesc[curSetIdx] ; //循环遍历参数集
    if ( NULL_var == CurrentSet ) break ;
    CATCkeListOfParm pListResult = CurrentSet->Parameters();

    .....

```

3.2.4 读取参数集中的参数

代码如下：

```

.....
for(int curParmIdx=1; curParmIdx<=pListResult->Size(); curParmIdx++)
//循环遍历一个参数集中的各个参数
{
    spBaseUknParm[curParmIdx] = (*pListResult)[curParmIdx] ;
    //获得参数
    rc = spBaseUknParm[curParmIdx]->QueryInterface
        (IID_CATICkeParm, (void**)&pCkeParam[curParmIdx]);

    .....

```

3.2.5 修改参数

代码如下：

```
.....  
    CATUnicodeString CurParmname = pCkeParam[curParmIdx]->Pathname();//获得参数名称  
    ShowString(CurParmname);  
    CATUnicodeString canshu = "1200mm";  
    pCkeParam[curParmIdx]->Valuate(canshu);////给参数赋值  
}  
} //结束后手动更新
```

4 使用设计表

4.1 创建和使用设计表

设计表允许设计者使用外部表格数据或 txt 数据来设计零件。例如将如下所示的 txt 文件用于零件设计：

Column1	Column2	Column3	Column4	Column5	Column6(deg)
7	7.5	Test design table	False	15mm	10
7.29	7	I hope it works	True	15	100

要将上述 txt 文件读取并使用必须经过以下几步：

- 1.采用手动或者自动的方式使设计表与 txt 文本文件链接。
- 2.获取联系。
- 3.显示和修改当前的配置。
- 4.修改配置后检查新参数值。

4.1.1 初始化

在进行下边内容之前初始化环境。初始化过程与创建公式的初始化完全相同。

初始化主要是指创建或打开一个文件，并获取它的根 Container，并且给 CATICkeParmFactory 接口赋值。关于 CATICkeParmFactory 接口必须注意两点，一是它可以采用任何根 container 实现，二是它可以提供任何参数和关系的创建方法。初始化代码如下所示：

```
HRESULT rc;  
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor(); //获得 Editor  
CATDocument *pDoc = pEditor->GetDocument(); //获得 Document  
if ( NULL ==pDoc ) return ;  
CATInit *pInitOnDoc = NULL ;  
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc); //从 doc 中获取 prt 文件的初始化  
if ( FAILED(rc) ) return ;
```

```

pInitOnDoc->Init(TRUE);
CATIContainerOfDocument_var spNewDocContainer = pDoc;
CATIContainer *cont = NULL;
rc = spNewDocContainer->GetSpecContainer(cont);//获取结构 Container
if (FAILED(rc) || cont == NULL)return ;
CATICkeParmFactory_var spFact = cont;

```

4.1.2 创建参数

此处采用 CATICkeParmFactory 的方法创建 6 个不同种类参数，代码如下所示：

```

CATICkeParm_var spPp1 = spFact->CreateInteger ("Column1",0);
CATICkeParm_var spPp2 = spFact->CreateReal ("r",0.0);
CATICkeParm_var spPp3 = spFact->CreateString ("s","");
CATICkeParm_var spPp4 = spFact->CreateBoolean ("b",CATCke::True);
CATICkeParm_var spPp5 = spFact->CreateLength ("l",0);
CATICkeParm_var spPp6 = spFact->CreateAngle ("a",0);

```

注意创建不同类型的参数时使用的方法不同。

4.1.3 创建设计表

采用 CATICkeParmFactory 中的创建设计表的方法 CreateDesignTable。

```

virtual CATIDesignTable_var CreateDesignTable(const CATUnicodeString& iRelationName ,
                                              const CATUnicodeString& iComment,
                                              const CATUnicodeString& iFilePath,
                                              intorientation,
                                              intsheetWithoutFile);

```

这个方法的第一个参数是设计表的 in 工厂，第二个参数是注释，第三个参数是设计表管关联的外部文件的路径。

如果第三个参数文件路径中既不包含 “.txt” 也不包含 “.xls” 或者 excel 应用不能启动的时候就会返回 NULL_var；否则 CreateDesignTable 必定会创建成功，即使创建一个空白文件。

4.1.4 设计表和参数关联

一般采用以下三步使设计表和参数关联。

第一步创建参数链表。

让链表自动和设计表中的值关联。这种情况只有参数值的名称和设计表的表头名称相同才能自动匹配成功。比如此处的 “Column1” 是设计表和参数共有的名称，只有 Column1 满足自动匹配条件，其他参数则自动匹配失败。自动匹配代码如下：

```

CATLISTV(CATBaseUnknown_var) spList;
spList.Append(spPp1);

```

第二步创建自动连接。

CATIDesignTable 接口中有自动创建连接方法 AutomaticAssociations。这个方法的第一个参数是 CATIParmPublisher 类型。根 container 就是一个 CATIParmPublisher。第二个参数采用 NULL_var 智能指针即可，第三个参数就是要连接到设计表的参数链表。代码如下所示：

```
spDesign->AutomaticAssociations(iCont, NULL_var, &spList);
```

第三步创建手动连接。

当用上述自动连接后可能会有部分连接成功，比如上述的 Column1 链接成功，而其他的链接都失败。此时就需要手动一个一个添加链接，添加链接采用 CATIDesignTable 接口的 AddAssociation 方法。AddAssociation 的第一个参数是设计表列的名称，第二参数是与设计表链接对应列链接的参数名称。手动链接代码如下所示：

```
spDesign->AddAssociation("Column2", spPp2);
```

```
spDesign->AddAssociation("Column3", spPp3);
```

```
spDesign->AddAssociation("Column4", spPp4);
```

```
spDesign->AddAssociation("Column5", spPp5);
```

AddAssociation 返回值为链接状态，状态值如下所示：NoError, ColumnNamesNotUnique, BadCellType, BadColumnType, NotAMagnitude, ParameterNotAssociated, AttributeProblem, BadConfiguration, ParseSheetError, ParameterAlreadyAssociated, ColumnAlreadyAssociated, ColumnDoesntExist。成链接功返回 0；没有对应的列返回 11。

4.1.5 获取链接数量

CATIDesignTable 接口的 Associations 方法可以获取为这个文件已经完成的链接数。代码如下所示：

```
int numb = spDesign->Associations()->Size();
```

这样就获取了链接的数量。一个是自动链接，四个是一一链接。

4.1.6 获取及设置当前配置

在设计表被创建并且其值与文件中的参数对应后是没有给定默认配置。设计表配置必须用 CATIDesignTable 接口中的 SetCurrentConfiguration 方法给定，参数为行号。使用 ConfigurationRow 可以获取配置的行号，若没有设置默认配置，则返回 0；代码如下所示：

```
int numb = spDesign->ConfigurationRow();//numb=0
```

```
spDesign->SetCurrentConfiguration(1);
```

```
numb= spDesign->ConfigurationRow();//numb=1
```

4.1.7 删除参数与设计表的联系

采用 CATIDesignTable 接口的 RemoveAssociation 方法可以删除联系。删除联系代码为：
spDesign->RemoveAssociation("Column4");

4.2 修改设计表的值

与设计表链接的文件操作的步骤一般为：首先获取结构 container(specification container)，然后给 CATICkeParmFactory 赋值，用 CATICkeParmFactory 创建设计表 CATIDesignTable，或者创建工作簿 CATICkeSheet 等。

若为读取 xls 文件，需要获取 Design table 在创建 Deign Table 的时候给定文件路径。然后用 CATIDesignTable::CellAsDouble 方法可以直接获取数值型内容，若获取其他类型的内容，如文本等需要借助 CATICkeSheet。从 Design Table 中获取 CATICkeSheet，在通过 CATICkeSheet::Cell() 等方法获取单元格中的内容。

若写入 xls 文件操作，则必须借助 CATICkeSheet。从 CATICkeParmFactory 创建工作簿 CATICkeSheet，给定保存文件的种类和路径，并用 CATICkeSheet::SetCell()方法即可将数据写入工作簿并保存。

对于操作 txt 文件的方法与 xls 相同。

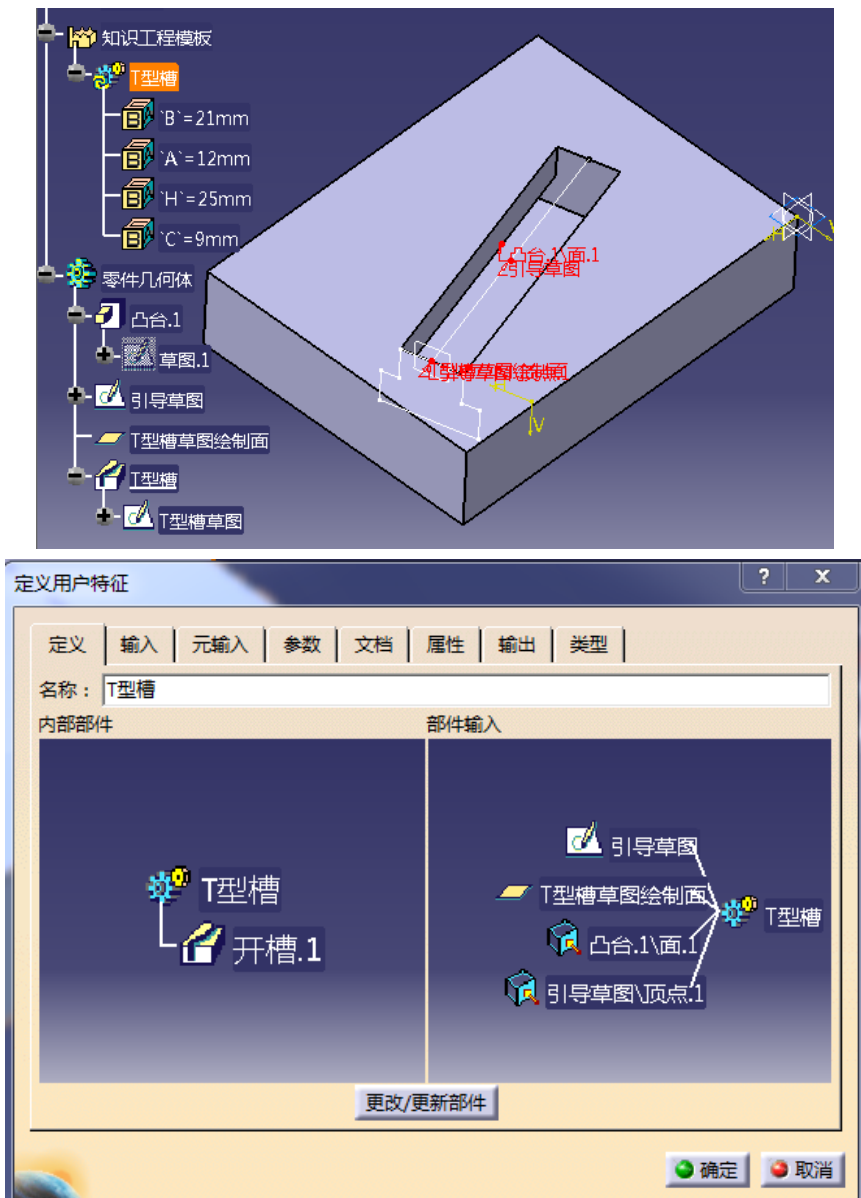
5 用户自定义特征

用户自定义特征包括用户特征和超级副本，它们的创建、实例化、修改过程相似。

5.1 定义

组件(component)；组件是形成用户自定义特征的参考特征。这些参考特征可以是一个几何特征，知识对象(规则，公式，设计表等)，约束，别的用户特征，曲面集(几何图形集和有序几何图形集)，一个几何体(非零件几何体)。一个超级副本或用户特征可以由任意多个组件构成，但这些组件必须来自同一个 Part 文件。

以绘制 T 型槽为例说明：T 型槽用户特征只有一个开槽组件。



输入 (Input); 输入是形成用户自定义特征时非直接选择的特征, 但是这个特征是一个组件和外部链接的特征。在用户自定义特征实例化是输入输入必须赋值。

以绘制 T 型槽为例说明输入。它的功能是选中一个引导线, 给定 T 型槽的尺寸参数, 沿着引导线绘制出 T 型槽。

共有 4 个输入, 引导草图 (引导草图)、T 型槽绘制平面 (引导草图)、T 型槽草图中心 (引导草图\顶点.1)、T 型槽放置面 (凸台.1\面.1)。

参数发布 (Published Parameter): 由上所述, 一个特征一旦被建立就包含有若干个已经发布的参数。用户特征的组件同样具有若干这样的参数, 在这些参数中有些可能在实例化过程中需要更改, 那么就需要提前选择出这样的参数, 这一选择过程可以看做发布参数 (Publish)。

例如实例化 T 型槽特征时可能需要修改 T 型槽的大小, 所以发布了草图中的 4 个长度尺寸。



5.2 创建用户自定义特征

5.2.1 获取用户自定义特征工厂

从 Part 文档获取结构 Container，之前将过获取方法有三种。这里采用文档初始化的方式。再从结构 Container 获取自定义特征工厂接口 CATIUdfFactory。代码如下：

```
CATInit *piInit = NULL ;
rc = pPartDocument->QueryInterface(IID_CATInit,(void **) &piInit);
CATIPrtContainer *piPartContainer = NULL ;
piPartContainer=(CATIPrtContainer*)piInit ->GetRootContainer("CATIPrtContainer");
CATIUdfFactory *piUdfFactoryOnPartContainer = NULL ;
piPartContainer->QueryInterface(IID_CATIUdfFactory,
                                (void **) &piUdfFactoryOnPartContainer);
```

其中 pPartDocument 是 Part 文件的 CATDocument 指针。

5.2.2 创建新特征参考

用自定义特征工厂创建新的特征参考。

对于创建一个用户特征代码如下：

```
CATUnicodeString UserFeatureReferenceName= "NameOfTheUserFeatureReference"
CATIUdfFeature_var spiUdfFeature = piUdfFactoryOnPartContainer
->CreateUserFeature(UserFeatureReferenceName);
```

其中 CreateUserFeature 方法返回了一个新的用户特征，参数是新用户特征的名称，这个名称在特征实例化的时候会使用到。如果没有给定名称，默认名称是用户特征的组件名称。

对于创建一个超级副本的代码如下：

```
CATIUdfFeature_var spiUdfFeature = piUdfFactoryOnPartContainer ->CreatePowerCopy();
CATIAlias * piAliasOnPowerCopy = NULL ;
```



```
spiUdfFeature->QueryInterface(IID_CATIAlias ,(void **) &piAliasOnPowerCopy);  
    CATUnicodeString PowerCopyReferenceName= "NameOfThePowerCopyReference"  
piAliasOnPowerCopy->SetAlias(PowerCopyReferenceName);
```

要设置名称超级副本的名称只能使用 CATIAlias 接口。

在一个 Part 文件中只能创建一个超级副本集或者一个知识工程模板集。

5.2.3 添加组件

形成用户特征或超级副本必须添加一个或多个组件，这些组件的特征组成一个链表，并由链表管理。代码如下：

```
CATListValCATBaseUnknown_var pComponentsList= new CATLISTV(CATBaseUnknown_var);  
pComponentsList->Append(spOnMyFirstFeature);  
pComponentsList->Append(spOnMySecondFeature);
```

必须注意的是 spOnMyFirstFeature 和 spOnMySecondFeature 是当前文档中特征的智能指针。

5.2.4 检查组件

用户特征或超级副本赋值（设置参数、输入等值）之前应该进行检查组件的步骤。代码如下：

```
rc = spiUdfFeature->VerifComponents(pComponentsList);
```

如果返回的结果是 S_OK，那么检查结果是成功的。

也可以选择在给用户特征赋值之后检查组件，但是这样的话，如果检查失败必须删除定义的用户特征重新定义一个新的用户特征。若在赋值之后检查，那么参数必须为 NULL。

5.2.5 组件赋值

CATIUdfFeature 提供了很多方法进行用户自定义特征的具体操作。

组件赋值的主要功能是从所选择的组件中确定用户特征的输入和参数。代码如下：

```
rc = spiUdfFeature->SetComponents(pComponentsList);
```

a.重命名输入

重命名输入不是必须进行的一项工作，但是为了让用户在使用用户特征时能够较好的理解输入的意义，还是应该重命名输入。

方法 GetInputsNumber 可以获取输入的数量，GetListInputs 可以获取具体的每个输入及输入的名称；再使用 SetInputRole 就可以重命名输入了。

b.发布需要的参数

组件中有很多参数，选择有用的参数将其发布出来。

通过 GetInternalParameters 将所有的内部参数获取出来，然后用 AddParameter 方法将其中具体某个参数发布，或者采用 RemoveParameter 方法将已经发布的参数取消发布。

GetParameters 方法可以返回已经发布的所有参数。发布参数的方法适用于用户自定义特征。

c.重命名发布的参数

重命名参数有利于用户特征实例化的时候便于理解。使用 SetParameterRole 方法可以重命名一个已经发布的参数。

5.2.6 添加新特征到结构树

对于用户特征代码为：

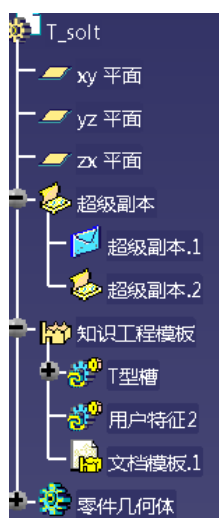
```
CATIUdfFeatureSet_var spiIdfFeatureSet = piUdfFactoryOnPartContainer->GetFeatureSet(1);  
spiIdfFeatureSet->Append(spiUdfFeature);
```

新的用户特征已经集成到知识工程模板集中了，知识工程模板集的位置为 1

对于超级副本代码为：

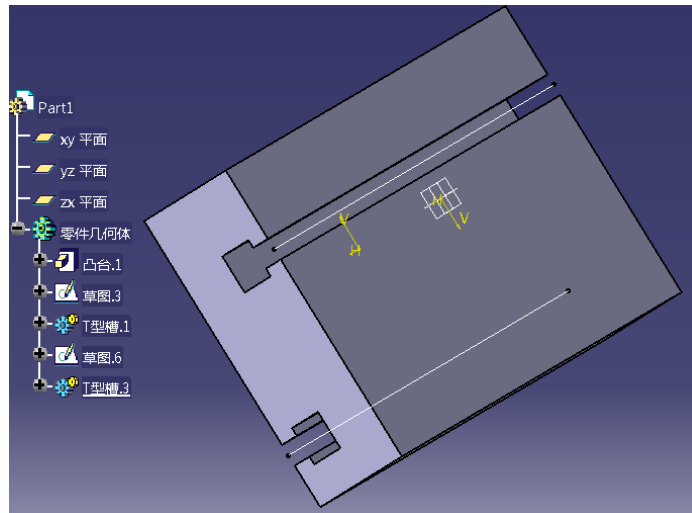
```
CATIUdfFeatureSet_var spiIdfFeatureSet = piUdfFactoryOnPartContainer->GetFeatureSet(0);  
spiIdfFeatureSet->Append(spiUdfFeature);
```

超级副本集的位置为 0;



5.3 用户自定义特征实例化

用户自定义特征在实例化中涉及引用 Part 文档和目标 Part 文档，引用 Part 文档是用户自定义特征创建的文档，而目标 Part 文档是用户自定义特征使用的位置。这两个文件最好不要是同一个文件。将 T 型槽实例化，如图所示：



5.3.1 获取要实例化的用户自定义特征

首先获取 Part 文档，获取结构 Container，从结构 Container 获取 CATIUdfFactory；

获取用户特征列表代码：

```
CATListValCATISpecObject_var *pFeatureList = NULL ;
```

```
pFeatureList = piUdfFactoryOnPartContainer ->GetUserFeatureList();
```

获取超级副本列表代码：

```
CATListValCATISpecObject_var *pFeatureList = NULL ;
```

```
pFeatureList = piUdfFactoryOnPartContainer ->GetPowerCopyList();
```

其中 piUdfFactoryOnPartContainer 就是 CATIUdfFactory 接口的指针。pFeatureList 是 CATISpecObject_var 的链表，链表中包含了可以实例化的用户特征或超级副本的参考。采用 CATIUdfInstantiate 接口可以将这些用户自定义特征实例化。代码如下所示：

```
CATISpecObject_var spFeatureReference = (*pFeatureList)[i] ;
```

```
CATIUdfInstantiate *piUdfInstantiateOnFeatRef = NULL ;
```

```
rc = spFeatureReference->QueryInterface(IID_CATIUdfInstantiate,
```

```
(void **) &piUdfInstantiateOnFeatRef);
```

5.3.2 设置目标位置

实例化的用户特征位置一定在几何特征集中，具体是在几何体、几何图形集还是有序几何图形集中，要由用户特征的本质确定。有两种方法可以确定实例化的具体位置。

1.如果在机械设计中包含多个几何特征集，实例化则出现在当前特征集中，使用 CATIPrtPart :: SetCurrentFeature 可以修改当前特征集。

具体代码如下所示：

```
CATISpecObject_var spSpecObjectOnMechanicalPart = piPartContainer->GetPart();
CATBaseUnknown_var spBuknOnMechanicalPart = spSpecObjectOnMechanicalPart ;
CATBaseUnknown * pMechanicalPart = (CATBaseUnknown *)spBuknOnMechanicalPart ;
CATPathElement PathOfTheMechanicalPart (pMechanicalPart);
CATBaseUnknown * pUIActiveObject = NULL ;
CATBaseUnknown_var spMechanicalPartDestination;
rc = piUdfInstantiateOnFeatRef->SetDestinationPath(&PathOfTheMechanicalPart,
pUIActiveObject,spMechanicalPartDestination);
```

2.指定一个目标文件中的特征并获取其位置。若选择了一个几何特征集，用户特征可以插入几何特征集内部，也可以在几何特征集后边。若选择了一个几何特征，用户特征只能在几何特征之后。选择的用户特征会自动设置为当前特征。

设置选择特征的路径并获取其位置的代码如下：

```
CATPathElement PathOfTheDestinationFeature (pDestinationFeature);
CATUnicodeString relativePosition = "After";
rc = piUdfInstantiateOnFeatRef->SetDestinationPathOfInsertion
(&PathOfTheDestinationFeature, relativePosition);
CATPathElement PathOfTheMechanicalPart (pMechanicalPart);
CATPathElement* PathOfTheDestinationFeature = NULL;
CATUnicodeString relativePosition;
rc = piUdfInstantiateOnFeatRef->GetDefaultDestinationOfInsertion
(&PathOfTheMechanicalPart,PathOfTheDestinationFeature,relativePosition);
```

5.3.3 给输入赋值

可以采用两种方式给输入赋值

第一种是采用 SetNewInput 方法给每一个输入逐个赋值，代码如下：

```
CATPathElement * pPathOnInput (pFeatureInput);
```

```
rc = piUdfInstantiateOnFeatRef->SetNewInput(x,pPathOnInput);
```

x 是一个输入在输入链表中的位置。采用 GetOldInputs 方法可以获取获取输入链表。

pPathOnInput 是特征的路径指针。

第二种是采用 UseIdenticalName 方法。具体代码如下：

```
rc = piUdfInstantiateOnFeatRef->UseIdenticalName(spRoot);
```

spRoot 在大多数情况下目标文档的智能指针。这个方法能够自动为每个输入查找特征，当 Part 文档中存在与输入名称相同的特征是匹配成功。一般在自动查找之后应该执行检查步骤。

5.3.4 给发布参数赋值

这一步骤不是必须要执行的，可以采用用户自定义特征在创建时的默认参数。采用

GetParameters 方法可以以链表的形式获取所有发布参数。这个方法同时可以获取参数规则列表，参数列表和规则列表的长度相同。参数赋值代码如下所示：

```
CATListValCATBaseUnknown_var * pParameterList = NULL;
CATListOfCATUnicodeString * pParameterRoleList = NULL ;
rc = piUdfInstantiateOnFeatRef->
GetParameters(pParameterList,pParameterRoleList);
if ( NULL != pParameterList )
{
int NbPublishedParameters = pParameterList->Size();
for ( int i= 1 ; i <= NbPublishedParameters ; i++ )
{
CATICkeParm_var spCkeParmOnParameter = (*pParameterList)[i];
if ( NULL_var != spCkeParmOnParameter )
{
//参数获取成功
}
}
}
```

可以采用 CATICkeParm 接口进行参数修改操作。

5.3.5 实例化

实例化代码如下：

```
rc = piUdfInstantiateOnFeatRef->Instantiate(NULL_var);
```

Instantiate 方法的参数必须为 NULL_var。

5.3.6 修改新用户特征的名称

修改名称只能对用户特征进行，使用 SetDisplayName 方法可以修改新用户特征实例的名称。具体代码如下：

```
CATUnicodeString NewNameOfTheInstance = "A different name " ;
rc = piUdfInstantiateOnFeatRef->SetDisplayName(NewNameOfTheInstance);
```

新用户特征实例的默认名称为用户特征创建时的名称。

5.3.7 结束实例化过程

在实例化完成后要进行结束实例化。

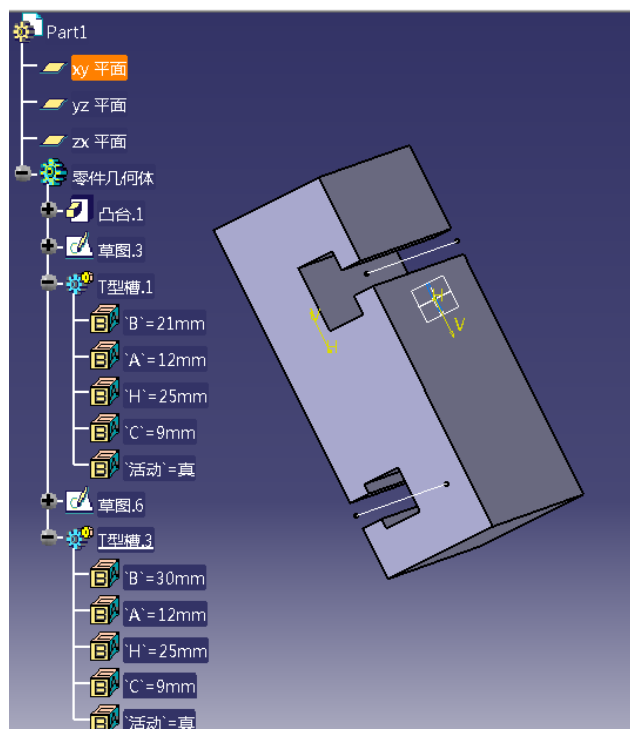
```
rc = piUdfInstantiateOnFeatRef->EndInstantiate();
```

在完成了结束实例化后可以进行下一个用户自定义特征实例化过程。

5.4 编辑用户自定义特征

在之前已经讲了如何创建用户自定义特征，如何将用户自定义特征实例化，在编辑用户自定义特征中主要讲如何读出已经实例化了的用户自定义特征，并改变其参数。

在给定了两个实例化的用户特征的模型中如何修改第二个实例化的用户特征。如图所示：修改第二个用户特征 T 型槽.3 的参数，如图所示：



5.4.1 获取用户特征

对于修改第一个用户特征和第二个用户特征而言，以下这段代码几乎相同，唯一的区别在于 CAAMcaGetGeometry 方法的第二个参数不同，它的第二个参数是用户特征的名称。代码如下所示：

```
CATBaseUnknown * pOnInstance = NULL ;  
    CATIUdfFeatureInstance * pIUdfFeatInstance = NULL ;  
    rc = ::CAAMcaGetGeometry(spSpecObjectCAAUdfModelPart,"T 型槽.3",&pOnInstance);  
    if ( FAILED(rc) ) return 1 ;  
    rc = pOnInstance->QueryInterface(IID_CATIUdfFeatureInstance,(void **) &pIUdfFeatInstance);  
    if ( FAILED(rc) ) return 1;
```

其中 spSpecObjectCAAUdfModelPart 是指向 Part 文件的 CATSpecObject 类型指针。

最后获取了 pIUdfFeatInstance 指针。

5.4.2 开始修改

这一步是在修改参数之前必须要执行的，只有调用了 `Init` 方法后才能修改实例化的用户特征。代码如下：

```
rc = pIUdfFeatInstance->Init();  
if ( FAILED(rc) ) return 1;
```

5.4.3 修改输入

对于用户特征 T 型槽而言有四个输入；修改第一个输入，T 型槽特征第一个输入为引导草图，将原来的引导草图草图.3 修改为草图.8，修改代码为：

```
CATBaseUnknown * pInput = NULL ;  
rc = ::CAAMcaGetGeometry(spSpecObjectCAAUdfModelPart,"草图.8",&pInput);  
if ( FAILED(rc) ) return 1 ;  
CATPathElement * pPathFirstInput = new CATPathElement(pInput);  
rc = pIUdfFeatInstance->SetNewInput(1,pPathFirstInput);  
if ( FAILED(rc) ) return 1 ;
```

草图.8 是在凸台表面的另外一个引导直线。首先获取草图.8 对象。采用的方法为 `CAAMcaGetGeometry`。在获取草图.8 对象的路径，再采用 `SetNewInput` 方法将草图.8 的路径设置为在 `pIUdfFeatInstance` 用户特征的第一个参数，其中 `SetNewInput` 的第一个参数 1 代表了第一个参数。

按照同样的方法可以修改第二个输入，第二个输入为绘制草图的中心点。设置中心点为点.1.代码如下：

```
CATBaseUnknown * pInput2 = NULL ;  
rc = ::CAAMcaGetGeometry(spSpecObjectCAAUdfModelPart,"点.1",&pInput2);  
if ( FAILED(rc) ) return 1 ;  
CATPathElement * pPathSndInput = new CATPathElement(pInput2);  
rc = pIUdfFeatInstance->SetNewInput(2, pPathSndInput);  
if ( FAILED(rc) ) return 1 ;
```

这样就修改了第二个输入为点.1

按照同样的方法修改第三、四个输入。

5.4.4 修改发布的参数

这步修改发布的参数值，修改发布参数和修改输入相互独立，完全可以只进行修改发布参数值。修改参数值的代码如下：

```
CATListValCATBaseUnknown_var * pListParam = NULL ;  
rc = pIUdfFeatInstance->GetParameters(pListParam) ;  
if ( FAILED(rc) ) return 1 ;  
CATICkeParm_var spCkeParm = (*pListParam)[1] ;
```

```

if ( NULL_var != spCkeParm)
{
spCkeParm->Valuate(0.050f);
}

```

采用 GetParameters 方法可以获得用户特征的发布参数列表。然后用专门修改参数的接口 CATICkeParm 将需要的值修改为想要的值。

5.4.5 修改结束

完成修改后要确认修改。代码如下：

```

rc = plUdfFeatInstance->Reset();
if ( FAILED(rc) ) return 1;

```

这一步是在修改用户特征实例之后必须执行的步骤。

6 知识工程功能实现

6.1 创建参数和公式

给按钮 001 添加响应函数代码：

```

HRESULT rc;
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor(); //获得Editor
CATDocument *pDoc = pEditor->GetDocument(); //获得Document
if ( NULL ==pDoc )return ;
CATInit *pInitOnDoc = NULL ;
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc); //从doc中获取prt文件的初始化
if ( FAILED(rc) )return;
pInitOnDoc->Init(TRUE);
CATContainerOfDocument_var spNewDocContainer = pDoc;
CATContainer *cont = NULL;
rc = spNewDocContainer->GetSpecContainer(cont); //获取结构container
if (FAILED(rc) || cont == NULL)return;
CATICkeParmFactory_var spFact = cont; //给参数工厂赋值
CATCkeListof(Parm) list; //声明要使用的参数列表
CATICkeParm_var spString1, spString2, spString3;
// find the factory
CATICkeParmFactory_var spParmFactory =spFact;
if (!!spParmFactory)
{
//创建参数
spString1 = spParmFactory->CreateString ("ModuleName", "CAALifRelations");
spString2 = spParmFactory->CreateString ("ProjectKey", "");
spString3 = spParmFactory->CreateString ("Topic", "Formulas");
}

```



```

CATIPrtContainer *pIPrtCont = NULL;//结构Container
rc = cont->QueryInterface(IID_CATIPrtContainer, (void**)&pIPrtCont);//
if(pIPrtCont==NULL) return;
CATISpecObject_var spSpecObjectOnPart = pIPrtCont->GetPart();//获取Part特征
    CATIPrtPart_var _spPart=NULL_var;
CATIPrtPart * pIPrtPart = NULL ;
    spSpecObjectOnPart->QueryInterface(IID_CATIPrtPart, (void**) & _spPart ) ;//
获取Part特征
    if(_spPart==NULL_var) return;
CATIParmPublisher * opiPublisher;//从Part特征获取发布集
rc = _spPart -> QueryInterface(IID_CATIParmPublisher,
(void**)&opiPublisher);
    CATISpecObject_var spParameterSet =
CATCkeGlobalFunctions::GetFunctionFactory()

    ->GetCurrentSet(CATICkeFunctionFactory::Parameter, opiPublisher, CATCke::True);
//设置发布集对象
    if(opiPublisher==NULL) return;
CATIParmPublisher_var spParmPublisher = spParameterSet;
    if (!!spParmPublisher)//将要发布的内容链接到发布集
    {
        spParmPublisher->Append(spString1);
        spParmPublisher->Append(spString2);
        spParmPublisher->Append(spString3);
    }
    //将创建的参数添加到链表
    list.Append (spString1);
    list.Append (spString2);
    list.Append (spString3);
//创建公式
    CATICkeRelation_var spFormula;
    spFormula = spParmFactory->CreateFormula ("ExtractForm","", "",
spString2, &list,
        "a1.Extract(0,6)", NULL_var, CATCke::False);
//发布创建的公式
    CATISpecObject_var spRelationSet =
CATCkeGlobalFunctions::GetFunctionFactory()

    ->GetCurrentSet(CATICkeFunctionFactory::Relation, opiPublisher, CATCke::True);
    CATIParmPublisher_var spParmPublisherRelation = spRelationSet;
    if (!!spParmPublisherRelation)
        spParmPublisherRelation->Append(spFormula);

```

```

}
// 显示结果, CAAlif 6个字母
CATUnicodeString key;
ExtractStringFromParm( spString2, key );
ShowString(key);

```

6.2 读取参数并赋值

在 002 按钮的回调函数中添加如下代码:

```

CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor(); //获得 Editor
CATDocument *pDoc = pEditor->GetDocument();                //获得 Document
if ( NULL ==pDoc )return ;
ValuatePara(pDoc);

```

其中 ValuatePara 函数是自定义函数, 需要自己编写声明和定义, ValuatePara 的定义如下所示:

```

void FormularDlg::ValuatePara(CATDocument* pPartDoc)
{
    HRESULT rc;
    CATBaseUnknown_var spBaseUknParm[20]={NULL};
    CATICkeParm* pCkeParam[20]={NULL};
    CATIPrtPart_var spPrtPart;//零件 part
    CATIContainerOfDocument_var spNewDocContainer = pPartDoc;
    CATIContainer *cont = NULL;
    rc = spNewDocContainer->GetSpecContainer(cont);//获取结构 container
    if (FAILED(rc) || cont == NULL)return;
    CATIPrtContainer *pIPrtCont = NULL;//结构 Container
    rc = cont->QueryInterface(IID_CATIPrtContainer, (void**)&pIPrtCont);//
    if(pIPrtCont==NULL) return;
    CATISpecObject_var spSpecObjectOnPart = pIPrtCont->GetPart();//获取 Part 特征
    spSpecObjectOnPart->QueryInterface(IID_CATIPrtPart,(void**) & spPrtPart );//获取 Part 特征
    if(spPrtPart==NULL_var) return;

    CATIDescendants *pPartAsDescendants = 0; //从 Part 特征获取继承对象
    rc = spPrtPart->QueryInterface(IID_CATIDescendants, (void**)&pPartAsDescendants);
    //获得参数集列表
    CATLISTV(CATISpecObject_var) ParmListDesc;
    pPartAsDescendants->GetAllChildren("CATICkeParameterSet", ParmListDesc);//找到参数集对象
    pPartAsDescendants->Release();
    pPartAsDescendants = NULL ;
    //获得每一个参数集中的参数
    for(int curSetIdx=1; curSetIdx<=ParmListDesc.Size(); curSetIdx++)
    {
        CATICkeParameterSet_var CurrentSet = ParmListDesc[curSetIdx] ;//循环遍历参数集
    }
}

```

```

        if ( NULL_var == CurrentSet ) break ;
        CATCkeListOfParm pListResult = CurrentSet->Parameters();
        for(int curParmIdx=1; curParmIdx<=pListResult->Size(); curParmIdx++)//循环遍历一个参数集中的各个
参数
    {
        spBaseUknParm[curParmIdx] = (*pListResult)[curParmIdx] ;
        if ( NULL_var == spBaseUknParm[curParmIdx] )
        {
            ShowString("FAILED Get CATBaseUnknown_var");
            continue;
        }
        //获得参数
        rc = spBaseUknParm[curParmIdx]->QueryInterface(IID_CATICkeParm,
(void**)&pCkeParam[curParmIdx]);
        if ( FAILED(rc) )
        {
            ShowString("FAILED Get CATICkeParm");
            continue;
        }
        CATUnicodeString CurParmname = pCkeParam[curParmIdx]->Pathname();//获得参数名称
        ShowString(CurParmname);
        CATUnicodeString canshu = "1200mm" ;
        pCkeParam[curParmIdx]->Valuate(canshu);////给参数赋值
    }
    }//结束后手动更新
}

```

6.3 使用设计表

在确定按钮的事件回调函数中添加代码：

```

HRESULT rc;
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor(); //获得Editor
CATDocument *pDoc = pEditor->GetDocument(); //获得Document
if ( NULL ==pDoc ) return ;
CATInit *pInitOnDoc = NULL ;
rc = pDoc->QueryInterface(IID_CATInit, (void**)&pInitOnDoc);//从doc中获取prt文
件的初始化
if ( FAILED(rc) ) return ;
pInitOnDoc->Init(TRUE);
CATIContainerOfDocument_var spNewDocContainer = pDoc;
CATIContainer *cont = NULL;
rc = spNewDocContainer->GetSpecContainer(cont);//获取结构Container
if (FAILED(rc) || cont == NULL) return ;

```

```
CATICkeParmFactory_var spFact = cont;
```

```
//创建参数
```

```
CATICkeParm_var spPp1 = spFact->CreateString ("Column1", "");  
CATICkeParm_var spPp2 = spFact->CreateString ("Column2", "");  
CATICkeParm_var spPp3 = spFact->CreateString ("Column3", "");  
CATICkeParm_var spPp4 = spFact->CreateString ("Column4", "");  
CATICkeParm_var spPp5 = spFact->CreateString ("Column5", "");  
CATICkeParm_var spPp6 = spFact->CreateString ("Column6", "");
```

```
//发布参数
```

```
CATIPrtContainer_var spPrtCont = cont;  
CATIParmPublisher_var spPubroot = NULL_var;
```

```
if (!!spPrtCont)
```

```
    spPubroot = spPrtCont->GetPart();
```

```
    CATISpecObject_var spParameterSet =
```

```
CATCkeGlobalFunctions::GetFunctionFactory()->GetCurrentSet(CATICkeFunctionFactory  
::Parameter, spPubroot, CATCke::True);
```

```
if (!!spParameterSet)
```

```
{
```

```
    CATIParmPublisher_var spParmPublisher = spParameterSet;
```

```
    if (!!spParmPublisher)
```

```
    {
```

```
        spParmPublisher->Append(spPp1);
```

```
        spParmPublisher->Append(spPp2);
```

```
        spParmPublisher->Append(spPp3);
```

```
        spParmPublisher->Append(spPp4);
```

```
        spParmPublisher->Append(spPp5);
```

```
        spParmPublisher->Append(spPp6);
```

```
    }
```

```
}
```

```
// 创建设计表
```

```
CATIDesignTable_var spDesign ;
```

```
spDesign = spFact->CreateDesignTable("DesignTable.1", "Design table test"
```

```
, "E:\\CATIA 二次开发\\catia 二次开发Train\\CATIA CAA开发\\Train_7 知识  
工程\\DesignTable.txt");
```

```
if(spDesign == NULL_var)
```

```
{
```

```
return ;
```

```
}
```

```
// 发布设计表
```

```
CATISpecObject_var spRelationSet =
```

```
CATCkeGlobalFunctions::GetFunctionFactory()->GetCurrentSet(CATICkeFunctionFactory  
::Relation, spPubroot, CATCke::True);
```

```

if (!!spRelationSet)
{
    CATIParmPublisher_var spParmPublisher = spRelationSet;
    if (!!spParmPublisher)
    {
        spParmPublisher->Append(spDesign);
    }
}
//创建设计表与参数的关联
// 自动关联
CATLISTV(CATBaseUnknown_var) list;
list.Append(spPp1);
list.Append(spPp2);
list.Append(spPp3);
list.Append(spPp4);
list.Append(spPp5);
list.Append(spPp6);
int nbAssoc = spDesign->AutomaticAssociations(cont, NULL_var, &list);
//手动关联
spDesign->AddAssociation("Column1", spPp1);
spDesign->AddAssociation("Column2", spPp2);
spDesign->AddAssociation("Column3", spPp3);
spDesign->AddAssociation("Column4", spPp4);
spDesign->AddAssociation("Column5", spPp5);
spDesign->AddAssociation("Column6", spPp6);
//设置配置的行号
spDesign->SetCurrentConfiguration(2);

```

7 额外的工程实例 P3

P3 工程要求对一个装配体进行参数化设计，通过修改各个零件的尺寸修改总装配体。

业务流程：打开参数化界面，修改参数，点击确定参数化装配体，保存已经参数化的装配体。

开发思路：@1 用手动完成装配体，保存在 Intel_a\UserDefined 文件夹下，作为参数化的模板；（建模时需要注意参数化的尺寸，添加约束时同样也要注意）

@2 获取界面上 Editor 中的尺寸；

@3 逐个参数化组成装配体的零件；

@3.1 打开零件文件；

@3.2 获取零件参数集；

@3.3 遍历参数并完成与界面中的参数名称匹配；

@3.4 采用界面中的值完成参数化；

@3.5 保存并关闭文件；

@4 参数化装配体（具体细节类似参数化零件）；

@5 将参数化完成的装配体添加到空的装配中（Product1）