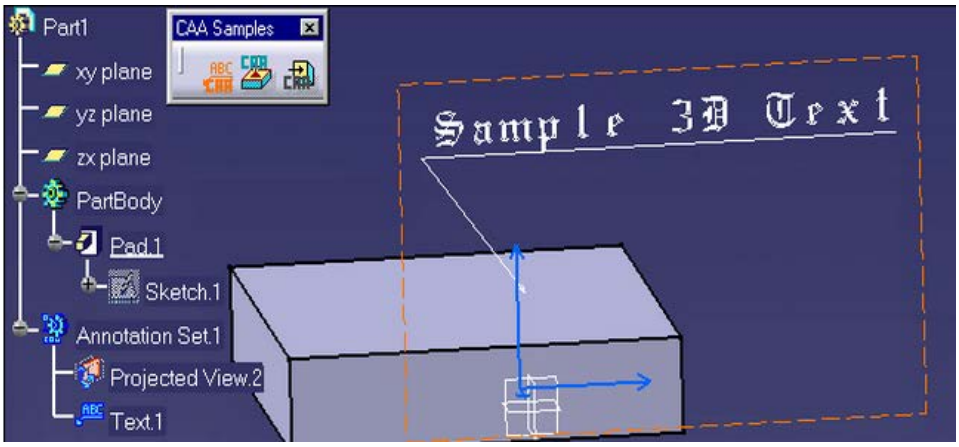
第 9 讲三维标注与公差

目录

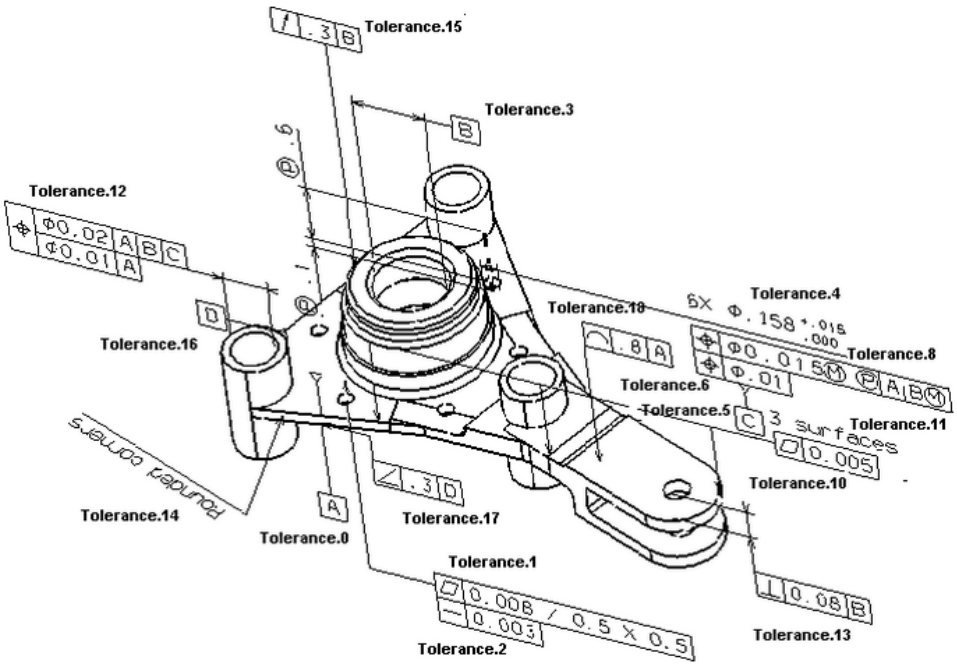
1 目标 2
2 从文件中获取标注集 3
2.1 获取文件 3
2.2 获取 TPS 文件 4
2.3 获取标注集 4
2.4 遍历标注集 4
2.5 遍历一个标注集中的各公差 5
2.6 获取公差的语义、关联的几何、公差带、尺寸界限 5
3 从标注获取几何元素并将其高亮 6
4 从选择的注释获取 TTRS 7
5 分析标注 10
5.1 分析公差语义 10
5.2 分析与公差关联的几何元素 16
5.3 分析公差带 17
5.4 分析尺寸界限 18
6 创建标注 22
6.1 在选定的平面上创建标注的时候采用的状态机 22
6.2 命令代理设置 22
6.3 开始创建标注 23
6.4 获取选择的几何体 23
6.5 修改文本的位置 24
6.6 修改文本的尺寸和字体 24
6.7 修改文本内容 25
7 源代码使用方法 27

1 目标

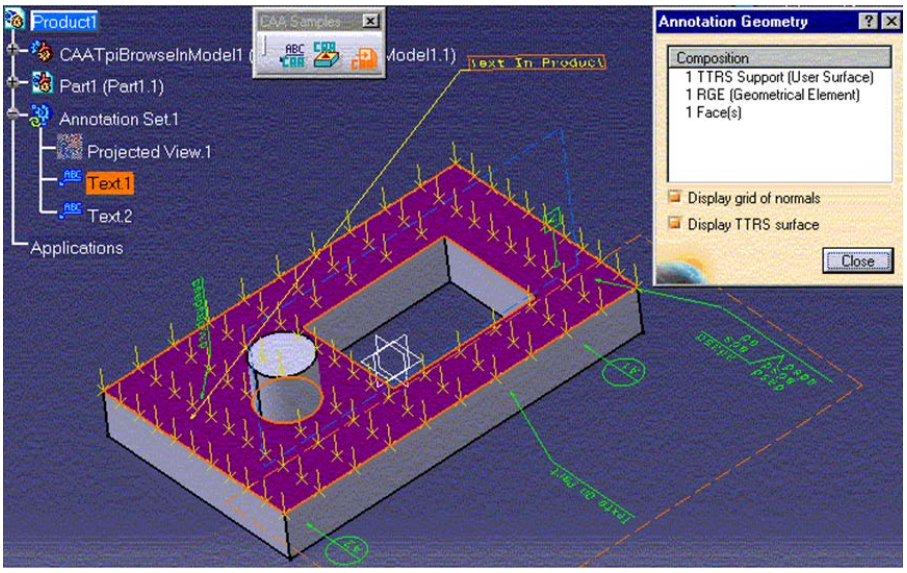
创建标注



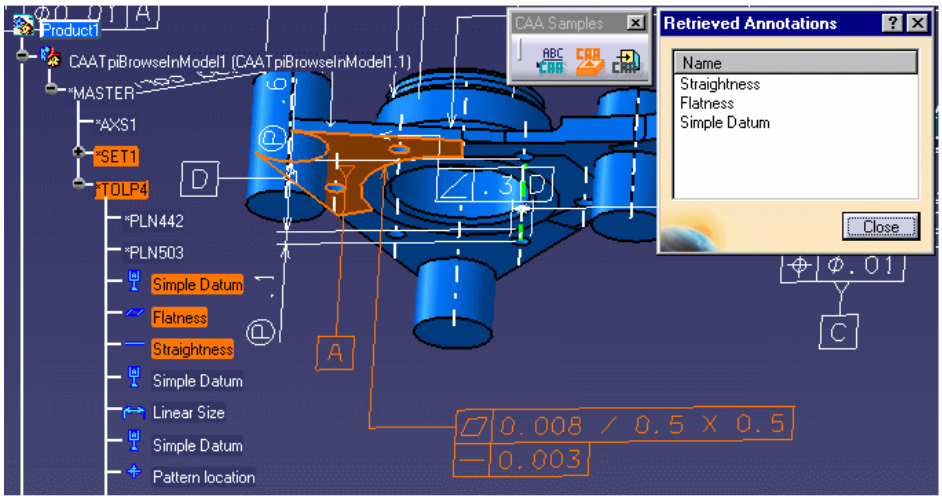
获取标注的信息



从标注中获取与之相关联的几何元素



从选取的几何元素获取与之关联的标注与公差



使用科技产品规范(TPS)接口。用例演示了 CATITPS 接口使用检索功能引用的一个 3d 注释。它还演示了如何扫描竞技场结构 CATITTRS 接口来检索 RGE(参考几何元素)。CATIRGE 和 CATIRGETopology 接口用于检索拓扑几何学的单元。最后,几何和拓扑评估执行创建和显示一个网格点与他们正常的单元。

2 从文件中获取标注集

2.1 获取文件

```
intAbstractEteamDlg::AbstractAll()
{
CATDocument* pCurDoc = BITFindCurrentDoc();
CATDocument* pDoc = pCurDoc;
```

```
char * TextPath="E:\\ooo.txt";
   FILE * pStream = fopen(TextPath, "w");
if ( pStream == NULL ) return (7);

   // Write copyright in stream
fprintf(pStream, "%s\n","COPYRIGHT DASSAULT SYSTEMES 2000");

   HRESULT rc = E_FAIL;
```

2.2 获取 TPS 文件

```
   // Retrieve TPS entry point interface on document//获取文件中 TPS 进入接口？？
CATITPSDocument * piTPSDoc = NULL;
if ( pDoc )
   {
rc = pDoc ->QueryInterface (IID_CATITPSDocument, (void**) &piTPSDoc);
   }
if ( FAILED(rc) ) return (8);
```

2.3 获取标注集

```
   // Retrieve list of tolerancing sets in document as a CATITPSList//获取文件中的标注集们？？
CATITPSList * piSetList = NULL;
rc = piTPSDoc ->GetSets(&piSetList);
piTPSDoc -> Release();
piTPSDoc = NULL;

if (!SUCCEEDED(rc) ) return 7;
      // Retrieve set count//获取标注集的数量
unsignedintSetCount = 0;
piSetList -> Count(&SetCount);

   // Iterate on list of set and for each set analyze tolerances//遍历每个标注集并分析公差
CATITPSSet* piSet = NULL;
```

2.4 遍历标注集

```
for( unsigned intSetIdx = 0; SetIdx<SetCount; SetIdx++ )
   {
         // Retrieve set in the list//从标注集链表中取出标注集 piCompOnSet 应该是一个标注集的一种数据表达
方式
CATITPSComponent* piCompOnSet = NULL;
rc = piSetList->Item (SetIdx, &piCompOnSet);
if ( !SUCCEEDED(rc) ) continue;
```

```
        // Analyze sets of piTPSDoc 分析文件中的标注集们
rc = piCompOnSet ->QueryInterface(IID_CATITPSSet, (void **)&piSet);
if ( !SUCCEEDED(rc) )    continue;
        // Retrieve tolerances that belong to set as a CATITPSList//获取一个标注集中的公差，公差以 CATITPSList 存储
CATITPSList* piToleList = NULL;
rc = piSet ->GetTPSs(&piToleList);
if ( !SUCCEEDED(rc) )    continue;
         // Retrieve tolerances count in set//获取一个标注集中的公差数量
unsignedintToleCount = 0;
piToleList    -> Count(&ToleCount);
```

2.5 遍历一个标注集中的各公差

```
         // Analyze tolerances of set//分析标注集中的公差
for ( unsigned intToleIdx = 0; ToleIdx<ToleCount; ToleIdx ++ )
          {
                  // Get ToleIdx tolerance in set 获取公差的 ID 和所在标注集的 ID
CATITPSComponent * piCompOnTole = NULL;
rc = piToleList -> Item (ToleIdx, &piCompOnTole);
if ( !SUCCEEDED(rc) ) continue;

fprintf (pStream,"%s%s%i%s%i%s\n",
                                          "------------------------ ",
" Set.", SetIdx,".Tolerance.",ToleIdx,
                                          " ------------------------");
```

2.6 获取公差的语义、关联的几何、公差带、尺寸界限

```
                    // Dump tolerance informations//获取公差信息
CAATpiDumpCATITPSSemanticValidity (piCompOnTole, pStream);
CAATpiDumpCATITPS                  (piCompOnTole, pStream);
CAATpiDumpCATITPSToleranceZone     (piCompOnTole, pStream);
CAATpiDumpCATITPSDimensionLimits   (piCompOnTole, pStream);
HighLAllAnn                        (piCompOnTole);//高亮全部标注...OK
PointLineFace                  (piCompOnTole);//高亮全部与标注相关的点、线、面


piCompOnTole -> Release();
                }
      }
piSetList -> Release();
   // Close output dump file
fclose(pStream);
```

return (0);
}

3 从标注获取几何元素并将其高亮

在上边代码中的自定义函数PointLineFace（）

```
//高亮全部与标注相关的点、线、面
void AbstractEteamDlg::PointLineFace(CATITPSComponent * ipiTole)
{
if ( !ipiTole )
  {
return;
  }
int TTRSNodeCount = 0;
int TTRSSupportCount = 0;
int RGECount = 0;
int FaceCount = 0;
int EdgeCount = 0;
int VertexCount = 0;
    CATFrmEditor *pEditor=CATFrmEditor::GetCurrentEditor();
    CATPathElement pUIPath = pEditor->GetUIActiveObject();
    CATPathElement * pPathTPS = &pUIPath;
    CATHSO * pHSO = pEditor -> GetHSO();

// Retrieve CATITPS interface on input tolerance
  CATITPS * piTPS = NULL;
  HRESULT rc = ipiTole -> QueryInterface (IID_CATITPS, (void **)&piTPS );
if ( SUCCEEDED(rc) )
  {
// Links to toleranced surfaces is retrieved as a TTRS list
unsignedint Count = 0;
    CATITTRSList * piTTRSList = NULL;
    CATITTRS * piTTRS = NULL;
    rc = piTPS -> GetTTRS (&piTTRSList);
if ( SUCCEEDED(rc) )
    {
// Dump TTRS count
      piTTRSList -> Count (&Count);
     CAT3DBagRep * pRep = new CAT3DBagRep();
// Iterate on TTRS list, and for each of them read
for ( unsignedint i = 0; i < Count; i++ )
```

```
      {// Retrieve TTRS number i in the list as a CATITTRS
        piTTRS = NULL;
        rc = piTTRSList -> Item (i, &piTTRS);
if ( SUCCEEDED(rc) )
        {
// Analyse TTRS Composition
          AnalyseTTRS (piTTRS, pHSO, pPathTPS,
                       TTRSNodeCount, TTRSSupportCount, RGECount,
                       FaceCount, EdgeCount, VertexCount, pRep);

// Construct a Rep to visualize TTRS and add it to ISO
//AddTTRSGeometryOnRepresentation (piTTRS, pPathTPS, pRep);

          piTTRS -> Release();
          piTTRS = NULL;
        }
      }
      piTTRSList -> Release();
    }
  }
}
```

4 从选择的注释获取 TTRS

当选择代理 OnAnnotationSelected 称为过渡方法。CATPathElement 选择 pPathTPS 检索通过调用 GetValue 选择代理。选中的 3 d 注释 piTPS 检索作为 CATITPS 接口指针使用 CATPathElement 的搜索方法。方法 CATITPS:GetTTRS 检索的列表竞技场直接由三维引用注释 CATITTRSList 接口指针。CATITTRSList::计数和 CATITTRSList::条目的方法允许列表上进行迭代。检索列表的每个元素作为 CATITTRS AnalyseTTRS 接口指针和提供的方法来获取一些信息组成。

```
booleanCAATpiAccessGeometryCmd::OnAnnotationSelected (void * ipData)
{
if ( !_pSelectionAgent || !_pPanel ) return (TRUE);

   HRESULT rc = E_FAIL;

intTTRSNodeCount = 0;
intTTRSSupportCount = 0;
intRGECount = 0;
intFaceCount = 0;
intEdgeCount = 0;
intVertexCount = 0;
```

```
   // Read display parameters from panel
   _pPanel ->GetRequiredDisplay (&_bDisplay3DGrid, &_bDisplayTTRSRep);

   // Retrieve the path of the selected annotation
CATPathElement * pPathTPS = _pSelectionAgent ->GetValue ();
if ( pPathTPS )
   {
      // Retrieve HSO from editor and empty it
CATFrmEditor * pEdt = GetEditor();
if ( pEdt )
      {
         CATHSO * pHSO = pEdt ->GetHSO();
if ( pHSO )
         {
pHSO -> Empty();
            // Add selected PathElement in the HSO, it will be highlighted
pHSO ->AddElements (pPathTPS);

            CATISO * pISO = pEdt ->GetISO();
if ( pISO )
            {
               // Clean existing element in ISO
pISO ->RemoveElement (_pModelObjectForAdditionalRep);

               // Retrieve CATITPS interface on selected 3D annotation
               CATITPS * piTPS = NULL;
rc = pPathTPS -> Search (IID_CATITPS, (void**) &piTPS);
if ( SUCCEEDED(rc) )
               {
                  // Retrieve the list of TTRSs which are directly referenced by
                  // the annotation, most often that list contains only 1 element,
                  // exeptions are Semantics Targets V5 and Default Annotation
CATITTRSList * piTTRSList = NULL;
rc = piTPS ->GetTTRS (&piTTRSList);
if ( SUCCEEDED(rc) )
                  {
unsignedintTTRSCount = 0;
piTTRSList -> Count (&TTRSCount);

                     CATITTRS * piTTRS = NULL;

                     // Allocate representation to display surfaces of TTRSs
                     // Points and normals on the TTRSs faces will be
                     // added in that Rep by AnalyseTTRS method.
```

```
                    CAT3DBagRep * pRep = new CAT3DBagRep();

                    // Iterate on the list TTRS
for ( unsigned intIdx = 0 ; Idx<TTRSCount ; Idx ++)
                    {
rc = piTTRSList -> Item (Idx, &piTTRS);
if ( SUCCEEDED(rc) )
                      {
                        // Analyse TTRS Composition
AnalyseTTRS (piTTRS, pHSO, pPathTPS,
TTRSNodeCount, TTRSSupportCount, RGECount,
FaceCount, EdgeCount, VertexCount, pRep);

                        // Construct a Rep to visualize TTRS and add it to ISO
AddTTRSGeometryOnRepresentation (piTTRS, pPathTPS, pRep);

piTTRS -> Release();
piTTRS = NULL;
                      }
                    }
                    // Add new Rep in ISO
                    _pModelObjectForAdditionalRep ->SetRep (pRep);

pISO ->AddElement (_pModelObjectForAdditionalRep);
pRep = NULL;

piTTRSList -> Release();
piTTRSList = NULL;
                  }
piTPS -> Release();
piTPS = NULL;
                }
pISO = NULL;
            }
            // No more elements to Add in the HSO, notification is send
            // and HSO content can be highlighted.
pHSO ->EndAddElements ();
pHSO = NULL;
          }
pEdt = NULL;
      }
pPathTPS = NULL;
   }
   ...
```

}

5 分析标注

对于标注的分析在程序源代码中使用的方法为：
CAATpiDumpCATITPSSemanticValidity (piCompOnTole, pStream);///分析语义（基准？形位公差？位置误差？）
CAATpiDumpCATITPS (piCompOnTole, pStream);//分析标注关联的几何元素形状（平面？圆柱面？）
CAATpiDumpCATITPSToleranceZone (piCompOnTole, pStream);//分析公差带（公差带的值）
CAATpiDumpCATITPSDimensionLimits (piCompOnTole, pStream);//分析尺寸界限（尺寸上下限值）

5.1 分析公差语义

```
voidAbstractEteamDlg::CAATpiDumpCATITPSSemanticValidity (CATITPSComponent * ipiTole,
                                                         FILE * ipStream)
{

if ( !ipiTole )
  {
return;
  }

  // Retrieve CATITPSSemanticValidity interface on input tolerance.
CATITPSSemanticValidity * piSemantic = NULL;
  HRESULT rc = ipiTole ->QueryInterface (IID_CATITPSSemanticValidity,
                                         (void **)&piSemantic);
if ( SUCCEEDED(rc) )
  {
    //-----------------------------------------------------------------------
    // Identify SuperType and Type
    //-----------------------------------------------------------------------

    // Retrieve tolerance SuperType as an IID
    IID * pSuperTypeAsIID = NULL;
piSemantic ->GetSuperType (&pSuperTypeAsIID);

    // Retrieve tolerance Type as an IID
    IID * pTypeAsIID = NULL;
piSemantic ->GetType (&pTypeAsIID);

    // Transform SuperType and Type IID in strings
CATUnicodeString Type ("Unknown");
```

```
CATUnicodeStringSuperType ("Unknown");

      // Form tolerances
      // If SuperType IID is the same that IID of CATITPSForm interface then this
      // tolerance belongs to the Form Super Type.
if ( CATCmpGuid(&IID_CATITPSForm, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Form";

         // If Type IID is the same that IID of CATITPSStraightness interface then
         // this tolerance is a Straightness tolerance.
if ( CATCmpGuid(&IID_CATITPSStraightness, pTypeAsIID) == TRUE )
            Type = "Straightness";
else if ( CATCmpGuid(&IID_CATITPSFlatness, pTypeAsIID) == TRUE )
            Type = "Flatness";
else if ( CATCmpGuid(&IID_CATITPSCircularity, pTypeAsIID) == TRUE )
            Type = "Circularity";
else if ( CATCmpGuid(&IID_CATITPSCylindricity, pTypeAsIID) == TRUE )
            Type = "Cylindricity";
else if ( CATCmpGuid(&IID_CATITPSProfileOfAnyLine, pTypeAsIID) == TRUE )
            Type = "Profile tolerance of any line";
else if ( CATCmpGuid(&IID_CATITPSProfileOfASurface, pTypeAsIID) == TRUE )
            Type = "Profile tolerance of any surface";
else if ( CATCmpGuid(&IID_CATITPSPatternTruePos, pTypeAsIID) == TRUE )
            Type = "Pattern localization";
      }
      // Orientation tolerances
else if ( CATCmpGuid(&IID_CATITPSOrientation, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Orientation";

if ( CATCmpGuid(&IID_CATITPSParallelism, pTypeAsIID) == TRUE )
            Type = "Parallelism";
else if ( CATCmpGuid(&IID_CATITPSPerpendicularity, pTypeAsIID) == TRUE )
            Type = "Perpendicularity";
else if ( CATCmpGuid(&IID_CATITPSAngularity, pTypeAsIID) == TRUE )
            Type = "Angularity";
      }
      // Position tolerances
else if ( CATCmpGuid(&IID_CATITPSPosition, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Position";

if ( CATCmpGuid(&IID_CATITPSTruePosition, pTypeAsIID) == TRUE )
```

```
            Type = "Positional";
else if ( CATCmpGuid(&IID_CATITPSConcentricity, pTypeAsIID) == TRUE )
            Type = "Concentricity";
else if ( CATCmpGuid(&IID_CATITPSSymmetry, pTypeAsIID) == TRUE )
            Type = "Symmetry";
else if ( CATCmpGuid(&IID_CATITPSPositionOfAnyLine, pTypeAsIID) == TRUE )
            Type = "Position linear profile";
else if ( CATCmpGuid(&IID_CATITPSPositionOfASurface, pTypeAsIID) == TRUE)
            Type = "Position surfacic profile";
      }
      // RunOut tolerances
else if ( CATCmpGuid(&IID_CATITPSRunOut, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Run-out";

if ( CATCmpGuid(&IID_CATITPSTotalRunOut, pTypeAsIID) == TRUE )
            Type = "Total run-out";
else if ( CATCmpGuid(&IID_CATITPSCircularRunOut, pTypeAsIID) == TRUE )
            Type = "Circular run-out";
      }
      // Dimension tolerances
else if ( CATCmpGuid(&IID_CATITPSDimension, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Size";

if ( CATCmpGuid(&IID_CATITPSLinearDimension, pTypeAsIID) == TRUE )
            Type = "Linear";
else if ( CATCmpGuid(&IID_CATITPSAngularDimension, pTypeAsIID) == TRUE )
            Type = "Angular";
else if ( CATCmpGuid(&IID_CATITPSChamferDimension, pTypeAsIID) == TRUE )
            Type = "Chamfer Dim";
      }
      // Datum tolerances
else if ( CATCmpGuid(&IID_CATITPSDatum, pSuperTypeAsIID) == TRUE )
      {
SuperType = "Datum";

if ( CATCmpGuid(&IID_CATITPSDatumSimple, pTypeAsIID) == TRUE )
           Type = "Simple datum";
else if ( CATCmpGuid(&IID_CATITPSDatumSystem, pTypeAsIID) == TRUE )
            Type = "Complex datum";
      }
      // NonSemantic tolerances
else if ( CATCmpGuid(&IID_CATITPSNonSemantic, pSuperTypeAsIID) == TRUE )
```

```
	{
SuperType = "Non Semantic";

if ( CATCmpGuid(&IID_CATITPSText, pTypeAsIID) == TRUE )
		Type = "Text";
else if ( CATCmpGuid(&IID_CATITPSFlagNote, pTypeAsIID) == TRUE )
		Type = "Flag Note";
	}
	// Dump Type and Super Type in output stream
fprintf(ipStream, "%-32s%s\n", "    SuperType:", SuperType.ConvertToChar());
fprintf(ipStream, "%-32s%s\n", "    Type:",          Type.ConvertToChar());

	//-----------------------------------------------------------------------
	// Access list of behavioral interfaces implemented on tolerance
	//-----------------------------------------------------------------------

	// Retrieve list of behavioral interfaces that must be analyzed for a
	// correct and plenty semantic understanding of this tolerance.
	// List is returned as a pointer on an array of pointers (IID adresses)
intSemanticItfCount = 0;
	IID ** ppSemanticItfList = NULL;

rc = piSemantic ->GetUnderstandingSemanticsItf (&SemanticItfCount,
&ppSemanticItfList);

	// First and second IID in list (idx 0,1) are SuperType and Type of
	// tolerance. Here start to iterate directly on index 2.
	// The list of semantic interfaces can contains interface IID that are not
	// published yet. In order to be protected against misunderstanding, the
	// client code must assume that, if one or some semantic interfaces are
	// unknown, all the interfaces and thus the whole tolerance must be
	// ignored.

charaInterfaceName [48];

for (intItfIdx = 2; ItfIdx<SemanticItfCount; ItfIdx ++ )
	{
	   // Initialize interface name
sprintf(aInterfaceName, "%s", "Unknown");

if ( ItfIdx == 2 )
	  {
fprintf(ipStream, "%-32s", "    Semantic behavioral itfs:");
	  }
```

```
else
        {
fprintf(ipStream, "%-32s", " ");
        }

        // Compare with CATITPSToleranceZone
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSToleranceZone) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSToleranceZone");
        }

        // Compare with CATITPSDimensionLimits
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSDimensionLimits) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSDimensionLimits");
        }

        // Compare with CATITPSMaterialCondition
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSMaterialCondition) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSMaterialCondition");
        }

        // Compare with CATITPSAssociatedRefFrame
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSAssociatedRefFrame) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSAssociatedRefFrame");
        }

        // Compare with CATITPSReferenceFrame
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSReferenceFrame) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSReferenceFrame");
        }

        // Compare with CATITPSEnvelopCondition
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSEnvelopCondition) == TRUE )
        {
```

```
sprintf(aInterfaceName, "%s", "CATITPSEnvelopCondition");
        }

        // Compare with CATITPSCompositeTolerance
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSCompositeTolerance) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSCompositeTolerance");
        }

        // Compare with CATITPSProjectedToleranceZone
if ( CATCmpGuid(ppSemanticItfList[ItfIdx],
&IID_CATITPSProjectedToleranceZone) == TRUE )
        {
sprintf(aInterfaceName, "%s", "CATITPSProjectedToleranceZone");
        }

fprintf(ipStream, "%s\n", aInterfaceName);
      }
delete [] ppSemanticItfList;

      //------------------------------------------------------------------------
      // Auto Diagnostic on semantic validity of tolerance
      //------------------------------------------------------------------------

      // Retrieve check diagnostic for this tolerance
wchar_t * pDiagnostic = NULL;
CATTPSStatus Status = CATTPSStatusUnknown;
rc = piSemantic -> Check(&pDiagnostic, &Status);

if ( SUCCEEDED(rc) )
{ // If Status is not OK, problem with this tolerance.
if ( Status != CATTPSStatusOK )
        {
           // if Check KO dump returned diagnostic
CATUnicodeStringDiag;
Diag.BuildFromWChar(pDiagnostic);
fprintf(ipStream, "%-32s%s\n",
"    Check Diagnostic:", Diag.ConvertToChar());
        }
delete [] pDiagnostic;
      }
piSemantic ->Release();
   }
```

```
}
```

5.2 分析与公差关联的几何元素

```
voidAbstractEteamDlg::CAATpiDumpCATITPS (CATITPSComponent * ipiTole,
                                         FILE * ipStream)
{

if ( !ipiTole )
   {
return;
   }

   // Retrieve CATITPS interface on input tolerance
   CATITPS * piTPS = NULL;
   HRESULT rc = ipiTole ->QueryInterface (IID_CATITPS, (void **)&piTPS );
if ( SUCCEEDED(rc) )
   {
      // Links to toleranced surfaces is retrieved as a TTRS list
unsignedint Count = 0;
CATITTRSList * piTTRSList = NULL;
      CATITTRS * piTTRS = NULL;
CATMmrTTRSClassTTRSClass = CATMmrUnknownTTRSClass;
      // Type CATMmrTTRSClass is an enum that describe functional surface class.
      // It can take values such as spherical, cylindrical, planar...
      // Unknown is a default value used for initialization purpose.

fprintf(ipStream, "%-32s", "    Link to surface:");

rc = piTPS ->GetTTRS (&piTTRSList);

if ( SUCCEEDED(rc) )
      {
         // Dump TTRS count
piTTRSList -> Count (&Count);
fprintf(ipStream, "%d%s", Count, " TTRS");

         // Iterate on TTRS list, and for each of them read
         // and dump his surface class
for ( unsigned int i = 0; i < Count; i++ )
         {
            // Retrieve TTRS number i in the list as a CATITTRS
piTTRS = NULL;
```

```
rc = piTTRSList -> Item (i, &piTTRS);
if ( SUCCEEDED(rc) )
              {
                 // Read TTRS class
TTRSClass = CATMmrUnknownTTRSClass;
piTTRS ->GetTTRSClass(TTRSClass);

                 // Dump TTRS class
fprintf(ipStream, '', '');
switch ( TTRSClass )
                 {
caseCATMmrSphericalTTRSClass:    fprintf(ipStream, "Spherical");break;
caseCATMmrCylindricalTTRSClass:fprintf(ipStream, "Cylindrical");
break;
caseCATMmrPlanarTTRSClass:         fprintf(ipStream, "Planar");      break;
caseCATMmrPrismaticTTRSClass:    fprintf(ipStream, "Prismatic");break;
caseCATMmrRevoluteTTRSClass:      fprintf(ipStream, "Revolute"); break;
caseCATMmrComplexTTRSClass:       fprintf(ipStream, "Complex");    break;
caseCATMmrUnknownTTRSClass:       fprintf(ipStream, "UnknownKO");break;
                 }
piTTRS -> Release();
              }
          }
piTTRSList -> Release();
       }
fprintf(ipStream, "\n");

piTPS ->Release();
   }
}
```

5.3 分析公差带

```
voidAbstractEteamDlg::CAATpiDumpCATITPSToleranceZone (CATITPSComponent * ipiTole,
                                                      FILE * ipStream)
{

if ( !ipiTole )
   {
return;
   }
   // Retrieve tolerance zone interface on input tolerance
CATITPSToleranceZone * piToleZoneOnTPS = NULL;
```

```
   HRESULT rc = ipiTole ->QueryInterface (IID_CATITPSToleranceZone,
                                                    (void **)&piToleZoneOnTPS);

   // QueryInterface succeeded only if CATITPSToleranceZone is a behavioral
   // interface of tolerance
if ( SUCCEEDED(rc) )
   {
      // Read tolerance zone value
doubleToleZoneValue = 0.0;
rc = piToleZoneOnTPS ->GetValue (&ToleZoneValue);
piToleZoneOnTPS ->Release();

      // Dump value to stream
fprintf(ipStream, "%-32s", "    Tolerance zone value:");
charaTolZoneAsString[24];
CAATpiSprintf(aTolZoneAsString,"%.3f", (float) ToleZoneValue);
fprintf(ipStream, "%s%s\n", aTolZoneAsString, " mm");
   }
}
```

5.4 分析尺寸界限

```
voidAbstractEteamDlg::CAATpiDumpCATITPSDimensionLimits (CATITPSComponent * ipiTole,
                                                    FILE * ipStream)
{

if ( !ipiTole )
   {
return;
   }

   // Retrieve dimension limits interface on input tolerance
CATITPSDimensionLimits * piDimLimitsOnTPS = NULL;
   HRESULT rc = ipiTole ->QueryInterface (IID_CATITPSDimensionLimits,
                                                    (void **)&piDimLimitsOnTPS);

   // QueryInterface succeeded only if CATITPSDimensionLimits is a behavioral
   // interface of tolerance. (Tolerance is a dimension)
if ( SUCCEEDED(rc) )
   {
      // Try to read up and bottom limits of the dimension
doubleUpLimit        = 0.0;
doubleBottomLimit = 0.0;
rc = piDimLimitsOnTPS ->GetLimits (&BottomLimit, &UpLimit);
```

```
      // HRESULT returned by GetLimits must be tested, because E_FAIL is returned
      // when, for instance, dimension is defined as "10 max". In that case, it
      // is a single limit dimension. Modifier on this dimension can be read by
      // using GetModifier method.
if ( SUCCEEDED(rc) )
      {
         // Dump up and bottom dimensions limits values
charaOneLimit[24];
CAATpiSprintf(aOneLimit,"%.3f", (float) UpLimit);
fprintf(ipStream, "%-32s%s%s\n", "    Up limit:", aOneLimit, " mm");
CAATpiSprintf(aOneLimit,"%.3f", (float) BottomLimit);
fprintf(ipStream, "%-32s%s%s\n", "    Bottom limit:", aOneLimit, " mm");
      }
else
      {
         // Read modifier on single limit dimension
CATTPSSingleLimit Modifier = CATTPSSLNotDefined;
rc = piDimLimitsOnTPS ->GetModifier (&Modifier);
fprintf(ipStream, "%-32s", "    Single Limit:");

         // Dump modifier
switch ( Modifier )
         {
caseCATTPSSLUnsupported:
fprintf(ipStream, "Unsupported");
break;
caseCATTPSSLNotDefined:
fprintf(ipStream, "Not Defined");
break;
caseCATTPSSLMaximum:
fprintf(ipStream, "Maximum");
break;
caseCATTPSSLMinimum:
fprintf(ipStream, "Minimum");
break;
caseCATTPSSLAsInformation:
fprintf(ipStream, "AsInformation");
break;
         }
fprintf(ipStream, "\n");
      }
piDimLimitsOnTPS -> Release();
piDimLimitsOnTPS = NULL;
```

```
   }
}

HRESULT CAATpiAccessGeometryCmd::AnalyseTTRS (CATITTRS * ipiTTRS,
                                              CATHSO * ipHSO,
CATPathElement * ipPathTPS,
int&oTTRSNodeCount,
int&oTTRSSupportCount,
int&oRGECount,
int&oFaceCount,
int&oEdgeCount,
int&oVertexCount,
                                              CAT3DBagRep * iopRep)
{
if ( ! ipiTTRS || !ipHSO || !ipPathTPS ) return (E_FAIL);

   HRESULT oRc = E_FAIL;

   // Retrieve TTRS nature: support or node.
CATMmrTTRSTypeTTRSType = ipiTTRS ->GetNature ();

   // Retrieve the components of the TTRS
CATLISTV(CATBaseUnknown_var) CompList;
   HRESULT rc = ipiTTRS ->GetComponents (CompList);
if ( SUCCEEDED(rc) )
   {
intComponentCount = CompList.Size();
intAnalyseSuccessCount = 0;

CATBaseUnknown_varspBaseComp;

      // If TTRS is a node, components are TTRS
if ( TTRSType == CATMmrNodeTTRS )
      {
oTTRSNodeCount ++; // Increment node count

         // Iterate on TTRS components and analyse them
         CATITTRS * piTTRSComp = NULL;
for ( int i = 1 ; i <= ComponentCount ; i++ )
         {
spBaseComp = CompList[i];
if ( NULL_var != spBaseComp )
            {
rc = spBaseComp ->QueryInterface(IID_CATITTRS, (void**)&piTTRSComp);
```

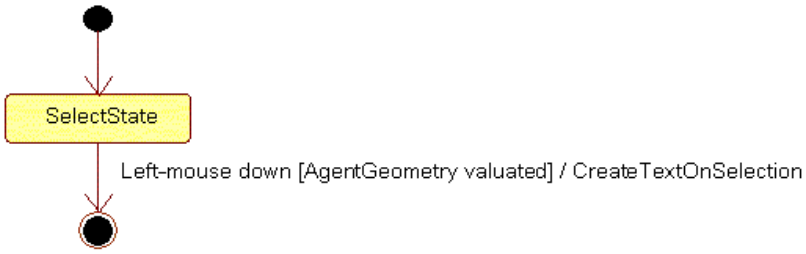
```
if ( SUCCEEDED(rc) )
                {
rc = AnalyseTTRS (piTTRSComp, ipHSO, ipPathTPS,
oTTRSNodeCount, oTTRSSupportCount, oRGECount,
oFaceCount, oEdgeCount, oVertexCount, iopRep);
if ( SUCCEEDED(rc) )
                  {
AnalyseSuccessCount++;
                  }
piTTRSComp -> Release();
piTTRSComp = NULL;
                }
spBaseComp = NULL_var;
            }
          }
      }
else // If TTRS is a support, components are RGE
      {
oTTRSSupportCount ++; // Increment support count

          // Iterate on RGE and analyse them
          CATIRGE * piRGE = NULL;
for ( int i = 1 ; i <= ComponentCount ; i++ )
          {
spBaseComp = CompList[i];
if ( NULL_var != spBaseComp )
            {
rc = spBaseComp ->QueryInterface (IID_CATIRGE, (void**)&piRGE);
if ( SUCCEEDED(rc) )
                {
rc = AnalyseRGE (piRGE, ipHSO, ipPathTPS,
oRGECount,
oFaceCount, oEdgeCount, oVertexCount, iopRep);
if ( SUCCEEDED(rc) )
                  {
AnalyseSuccessCount++;
                  }
piRGE -> Release();
piRGE = NULL;
                }
spBaseComp = NULL_var;
            }
          }
      }
```

```
      // method return S_OK only if each component is analysedsuccessfuly
if ( ComponentCount == AnalyseSuccessCount )
      {
oRc = S_OK;
      }
   }
return (oRc);
}
```

6 创建标注

6.1 在选定的平面上创建标注的时候采用的状态机



6.2 命令代理设置

```
voidCAATpiCreateTextCmd::BuildGraph ()
{
   // Create selection agent
   _pAgentGeometry = new CATPathElementAgent ("AgentGeometry",
                                                        NULL,
CATDlgEngWithPrevaluation|
CATDlgEngMultiAcquisition|
CATDlgEngWithPSOHSO);
CATListOfCATStringTypeList;

   // Retrieve CATITPSFactoryTTRS interfaces
CATITPSFactoryTTRS * piFactTTRS = NULL;
   HRESULT rc = CATTPSInstantiateComponent (DfTPS_ItfTPSFactoryTTRS,
                                                       (void**) &piFactTTRS);
if ( SUCCEEDED(rc) )
   {
      // Obtain Filter that must be used for selecting geometry
      // to create 3D annotation.
piFactTTRS ->ObtainOrderedTypeList (TypeList);
```

```
piFactTTRS -> Release();
piFactTTRS = NULL;
  }
  _pAgentGeometry ->SetOrderedTypeList(TypeList);

AddCSOClient (_pAgentGeometry);

  // Create state
CATDialogState * pSelectState = GetInitialState("SelectState");
if ( pSelectState )
  {
    // Plug selection agent
pSelectState ->AddDialogAgent(_pAgentGeometry);

    // Define transitions
AddTransition (pSelectState,
                          NULL,
IsOutputSetCondition(_pAgentGeometry),
Action((ActionMethod)&
CAATpiCreateTextCmd::CreateTextOnSelection));
  }
}
```

6.3 开始创建标注

```
boolean CAATpiCreateTextCmd::CreateTextOnSelection (void * ipData)
{
if ( !_pAgentGeometry ) return (TRUE);

  HRESULT rc = E_FAIL;
```

6.4 获取选择的几何体

```
  CATSO * pSelection = _pAgentGeometry -> GetListOfValues();
if ( pSelection )
  {
// 获取CATITPSFactoryAdvanced 接口
    CATITPSFactoryAdvanced * piFactAdv = NULL;
    rc = CATTPSInstantiateComponent (DfTPS_ItfTPSFactoryAdvanced,
                                     (void**) & piFactAdv);
if ( SUCCEEDED(rc) )
    {
```

```
      CATITPSText * piText = NULL;
CATUnicodeString TextString("Sample 3D Text");
      CATMathPlane Plane = CATMathOIJ;
      rc = piFactAdv -> CreateTextOnGeometry (pSelection, &Plane,
&TextString , &piText);
if ( SUCCEEDED(rc) )
      {
```

6.5 修改文本的位置

```
        CATIDrwAnnotation * piAnnot = NULL;
        rc = piText -> QueryInterface (IID_CATIDrwAnnotation,
                                       (void**) & piAnnot);
if ( SUCCEEDED(rc) )
        {
double DeltaX = -20.0;
double DeltaY = +20;

          piAnnot -> Move (DeltaX, DeltaY);

          piAnnot -> Release();
          piAnnot = NULL;
        }
```

6.6 修改文本的尺寸和字体

```
        CATIDrwTextProperties * piTxtProp = NULL;
        rc = piText -> QueryInterface (IID_CATIDrwTextProperties,
                                       (void**) & piTxtProp);
if ( SUCCEEDED(rc) )
        {
//修改字体为mm
          piTxtProp -> SetFontSize (7.0);

// 使用Gothic字体
CATUnicodeString FontName("GOTH");
          piTxtProp -> SetFontName(FontName);

          piTxtProp -> Release();
          piTxtProp = NULL;
        }

// Change Leader Extremity Symbol to a Filled Circle
```

```
        CATIDrwEltWithLeader * piEltWithLeader = NULL;
        rc = piText -> QueryInterface(IID_CATIDrwEltWithLeader,
                                      (void **) &piEltWithLeader);
if ( SUCCEEDED(rc) )
        {
int LeaderCount = piEltWithLeader -> GetNbLeader ();
if ( LeaderCount >= 1 )
          {
            CATIDrwLeader_var spDrwLeader = piEltWithLeader -> GetLeader (1);
if ( NULL_var != spDrwLeader )
            {
// FILLED_CIRCLE is a symbol type defined in CATSymbolType.h
int SymbType = FILLED_CIRCLE;
              spDrwLeader -> SetSymbolType (SymbType);
            }
          }
          piEltWithLeader -> Release();
          piEltWithLeader = NULL;
        }
```

6.7 修改文本内容

```
CATUnicodeString NewText("Sample 3D Text !!!");
wchar_t * pString  = newwchar_t [1 + NewText.GetLengthInChar ()];

        TextString.ConvertToWChar (pString);
        piText -> SetText (pString);

delete [] pString;
        pString = NULL;

        piText -> GetText (&pString);
CATUnicodeString ReadText;
        ReadText.BuildFromWChar(pString);

delete [] pString;
        pString = NULL;

// Use CATIDrwTextProperties::Refresh for updating visualization
// after leader and text modification
        rc = piText -> QueryInterface (IID_CATIDrwTextProperties,
                                       (void**) & piTxtProp);
if ( SUCCEEDED(rc) )
```

```
        {
          piTxtProp -> Refresh();
          piTxtProp -> Release();
          piTxtProp = NULL;
        }

        CATITPS * piTPS = NULL;
        rc = piText -> QueryInterface (IID_CATITPS, (void**)&piTPS);
if ( SUCCEEDED(rc) )
        {
          CATITPSText * piTxt = NULL;
CATUnicodeString Txt("Text on a Text");
          rc = piFactAdv -> CreateTextOnAnnotation (piTPS, &Txt, &piTxt);
if ( SUCCEEDED(rc) )
          {

// Modifying Text Size And Font
            CATIDrwTextProperties * piTxtProp = NULL;
            rc = piTxt -> QueryInterface (IID_CATIDrwTextProperties,
                                          (void**) & piTxtProp);
if ( SUCCEEDED(rc) )
            {
// Change Font Size to 7.0 millimeters
              piTxtProp -> SetFontSize (7.0);

// Use Gothic Font
CATUnicodeString FontName("GOTH");
              piTxtProp -> SetFontName(FontName);
              piTxtProp -> Refresh();
              piTxtProp -> Release();
              piTxtProp = NULL;
            }

            piTxt -> Release();
            piTxt = NULL;
          }
          piTPS -> Release();
          piTPS = NULL;
        }
        piText -> Release();
        piText = NULL;
      }
      piFactAdv -> Release();
      piFactAdv = NULL;
```

```
    }
  }
return (TRUE);
}
```

7 源代码使用方法

Train 9 的工程实例为 p34_ALiClampSolution_Anno；其中 AbstractEteamDlg 为主要实现标注提取，高亮、分析标注等相关功能。CAATpiCreateTextCmd 主要实现创建标注、修改字体字号等功能。