

---


# 第一讲：CAA 简介和架构

## 目录

1	综述.....	1
1.1	CATIA 二次开发简介 .....	1
1.1.1	VB 语言开发 .....	5
1.1.2	CAA 开发 .....	5
1.1.3	CAA 架构 .....	6
1.1.4	帮助资料介绍.....	8
1.2	CATIA 二次开发架构.....	10
1.3	CATIA 主界面概述.....	11
2	工程创建.....	13
2.1	添加工具条和菜单.....	13
2.1.1	新建工作空间（workspace）和框架（frame） .....	13
2.1.2	添加模块（Module）和组件（component） .....	18
2.1.3	添加相应代码.....	23
2.1.4	CATNIs 和 CATRsc 的使用 .....	27
2.2	新建命令（Command）和对话框（Dialog） .....	30
2.2.1	创建新的模块（module） .....	30
2.2.2	插入命令（command） .....	32
2.2.3	加入新的对话框（Dialog） .....	33
2.2.4	为控价添加回调函数（Callback Wizard） .....	38
2.3	实例运用.....	39
2.3.1	对话框简介.....	39
2.3.2	读取 txt 文件 .....	40
2.3.1	创建几何特征（Point）和实体特征（Solid） .....	42
3	编译和调试.....	44
4	复制工作空间.....	44
5	总结： .....	47

## 1 综述

### 1.1 CATIA 二次开发简介

Component Application Architecture (CAA) 组件应用架构，是 Dassault Systemes 产品扩展和客户进行二次开发的强有力的工具。以  为商标的 Dassault Systemes 已形成六大支柱

产品，通过 PPR HUB 进行集成，对产品的生命周期进行全方位管理。

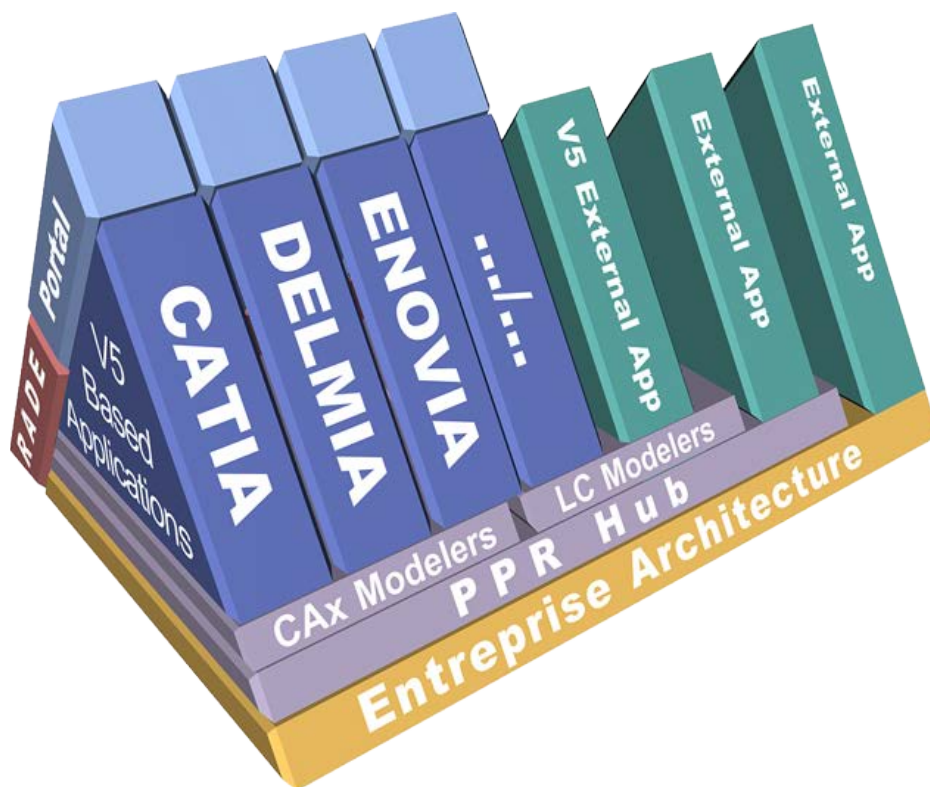


图 1 Dassault Systemes 产品结构逻辑关系

这六大产品包括：

CATIA V5：产品的数字化设计和数字样机技术。

ENOVIA：产品数据管理和协同工作环境。

DELMIA：数字化工厂，包括工艺设计、资源控制、工厂布置和生产模拟等。

另外还有 SOLIDWORKS、SMARTTEAM 也可进行数字化设计和管理，由用户情况决定。

特别要提的是 SPATIAL，专门进行 CAA 架构的开发和研究。

PPR HUB 是 Dassault Systemes 3D 产品生命周期管理（Product Lifecycle Management ,PLM）解决方案的核心，确保 CATIA、ENOVIA、DELMIA 三者之间的整合。

Dassault Systemes 的这套解决方案得利于开放式，可扩展的模块化开发架构 C A A，使得全球诸多开发商可以参与 Dassault Systemes 的研发。

对客户而言，CAA 可以进行从简单到复杂的二次开发工作，而且和原系统的结合非常紧密，如果没有特别的说明，无法把客户所研发的功能从原系统中区分出来，这非常有利于用户的使用和集成。

---

CAA 的实现，是通过提供的快速应用研发环境 RADE 和不同的 API 接口程序来完成的。

快速应用研发环境 Rapid Application Development Environment(RADE)是一个可视化的集成开发环境，它提供完整的编程工具组。实际上 R A D E 以 Microsoft Visual Studio VC++为载体，在 VC++环境中增加了 CAA 的开发工具。

API 提供了操作各种对象的方法、工具和接口。

- Dassault Systemes 提供的 CAA 产品包括如下内容：
- CAA RADE 快速开发环境，基于 Microsoft Visual Studio
- CAA CATIA V5 API，CATIA V5 应用开发工具
- CAA ENOVIA LCA V5 API，ENOVIA LCA 应用开发工具
- CAA DELMIA V5 API，DELMIA V5 应用开发工具
- CAA ENOVIA PORTAL V5 API，ENOVIA PORTAL V5 应用开发工具

CATIA 是 Computer Aided Tri-dimensional Interface Application 的缩写，是世界第一大飞机制造公司法国达索公司的产品，是世界上主流的 CAD/CAE/CAM 一体化集成软件。CATIA 开放了大部分接口，提供了强大的二次开发平台，用于满足用户对软件功能和性能的要求和软件本地化、用户化的要求。CATIA 的二次开发接口主要是通过进程内和进程外两种方式与外部程序通信。CATIA 软件与脚本运行在同一进程地址空间，即为进程内应用程序，比如宏 (Macro) 方式；CATIA 与外部程序在不同进程地址空间运行为进程外应用程序。

具体来说，CATIA 主要有两种二次开发方法：使用宏对 CATIA 进行二次开发和使用组件应用架构 (CAA-RADE) 对 CATIA 进行二次开发。相比较而言，CAA 方法具有强大的交互、集成功能，可以实现深层次和复杂系统的开发，有良好的开放性和稳定性，且 CAA 方法综合了 C++ 本身丰富的库，具有强大的界面开发功能，本文采用 CAA 方法对 CATIA 进行开发。

CAA (Components Application Architecture) 是组件应用架构技术，是达索系统产品扩展和客户进行二次开发的强有力工具，具有强大的集成、交互功能。如图 2 所示，在 CAA 中，组件 (Compenent) 是软件的基本组成部分，主要指可重用对象，代表了诸如点、线、面等对象；接口则代表实现组件的类型和行为，由一组对应于组件的操作集组成。应用程序通过接口与组件发生关系，获取一个指向接口的指针，调用接口所提供的方法，对组件进行相应的操作，完成程序所要求的功能。

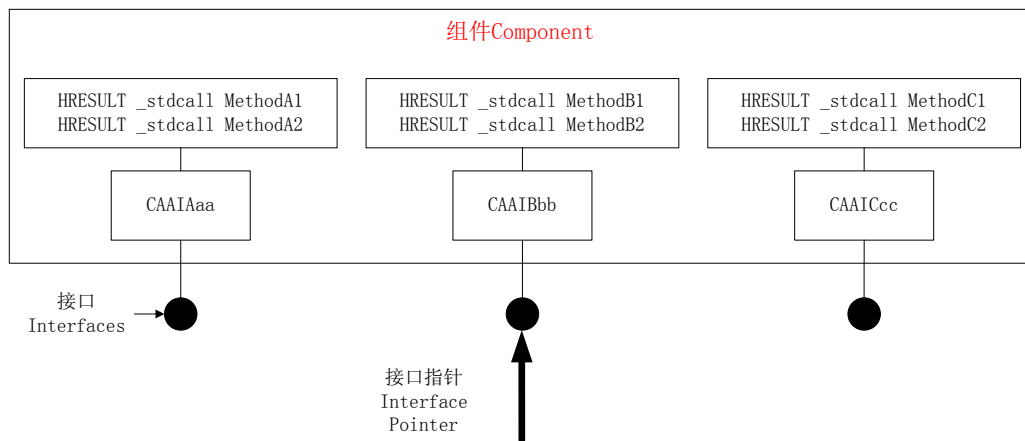


图 2 CAA 组件和接口

CATIA V5 采用基于 COM（Components Object Model，组件对象模型）技术的自动化提供二次开发接口。COM 是 CATIA V5 中自动化接口的核心技术，可以实现不同的组件在二进制可执行代码级基础上相互通信，并提供对外访问接口。如图 3 所示，IUnknown 接口是 COM 的基接口，COM 的每一个接口都是由基接口 IUnknown 继承，CATBaseUnknown 是 CAA 提供的 CATIA V5 所有接口的基类，AnyObject、Collection、Reference 是其它对象的抽象基类。AnyObject 是 Application、Window、Document 等对象的基类，Collection 是 Windows、Document、Products 等对象的基类，Reference 用于表示指向另一个对象的对象。CATIA V5 软件本身的开发即是遵循面向对象（COM）的设计思想，构建了基于组件的体系结构。

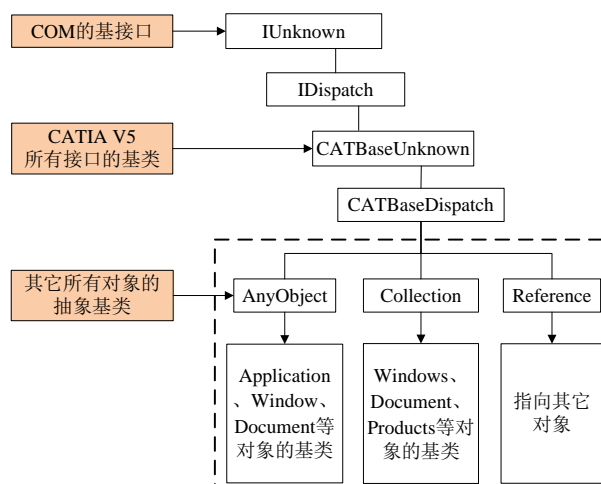


图 3 CATIA V5 的 COM 模型

在每个 CAA 中，至少有一个框架（framework），每个框架中至少有一个模块（module）。在每一个框架中包含一个 IdentityCard.h 文件，当此框架与其他框架进行联系时，通过此文件中的宏“AddPrereqComponent(framework,protected)”进行声明。

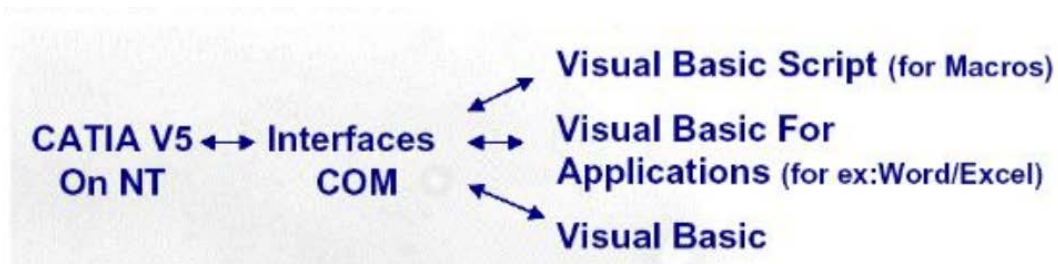
通过上述方法对 CATIA V5 组件的操作，用户可以组装 CATIA 自身的 CAA 组件实现对 CATIA 的功能扩展，也可以通过开发 CAA 组件实现自定义功能，从而实现系统的维护、管理

和扩展。

由于 CATIA 本身的 CAE 功能不够强大，只能进行简单的结构分析和模态等分析，不能满足成形分析等复杂的非线性分析，目前大部分成形性仿真分析都是在专业的有限元分析软件中进行，或者利用 CATIA 强大的二次开发功能，开发适用于具体问题的数值算法。

### 1.1.1 VB 语言开发

VB 语言开发 CATIA 有三种方法，Visual Basic Script for Macros (VBScript)、Visual Basic for Application (VBA)、Visual Basic。



Visual Basic(Visual Basic 专业版):

VB 是个完全的版本。1. 可以编制独立的程序。2. 也可以创建 ActiveX 和程序服务器。3. 可以被编译。4. VB 提供了自己的文档——《在线帮助》

VBA 是 VB 的一个子集。1. VBA 是程序中的主机，就象 Word,Excel。

2. 它提供了一个带有编辑器、除错器、帮助、对象浏览器、完成器的完整的开发环境。3. 利用它的 Tools-Reference 菜单可以声明使用的对象库。这在完成和对象浏览器之间建立了一个通道。

VBScript(Visual Basic Script):

VBScript 是 VB 的一个子集。1. 更简单的 VB 解释语言。2. 可以调运 CATIA 对象。3. 没有数据类型的区别。系统总是动态的调用函数和对象属性。(迟绑定调用对象通过他们的 Idispach 接口)。CATIA 录制宏和脚本就属于这种开发方式。

### 1.1.2 CAA 开发

CAA 全称 Component Application Architecture，组件应用架构。

CATIA 本身按照组件模型建立，用户可以开发自己的 CAA 组件，对 CATIA V5 进行扩展；也可以把用户自己开发的 CAA 组件结合起来。实现用户自定义的应用。

CAA 是 CATIA 的一整套 C++ 函数库，该函数库在 CATIA 运行时加载。用户通过安装 RADE (Rapid Application Development Environment) 模块，用户可以在 VC++ 编程环境下编制程序，与 CATIA 进行通信，从而对 CATIA 进行二次开发。

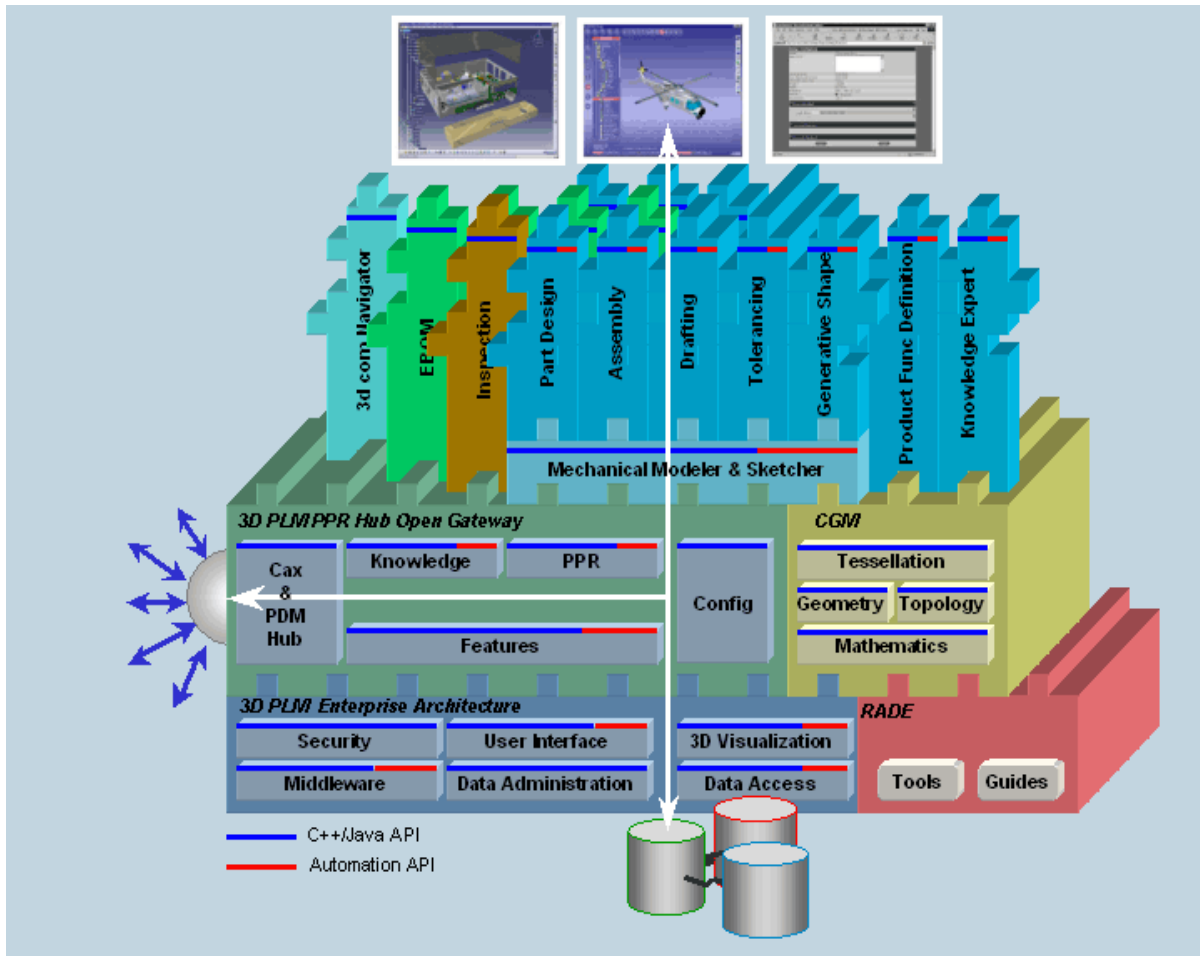
---

RADE 是快速应用研发环境，是一个可视化的集成开发环境，提供完成的编程工具组。RADE 以 Microsoft Visual C++为载体，开发工具完全集成在 VC++环境中。RADE 包括一些列工具。TCK (tool configuration key)、MAB (多空间应用生成器)、CUT (C++单元调试管理器)、MKMK (CAA 编译工具)、CID (C++交互式面板)。

### 1.1.3 CAA 架构

在软件开发领域，使用面向对象的程序设计 (OOP:Object-Oriented-Programming) 已经成为软件开发设计的主流，它为软件的发展带来了很多好处：可复用性、抽象性、封装性等。软件专家把软件开发看作是对象的组合。

面向对象程序设计是一种程序设计方法，而建立在面向对象程序设计基础之上的组件对象模型 (COM) 和对象的连接和嵌入 (OLE) 技术，使程序设计更加容易且趋于标准化，使程序的使用更加简洁明了。Microsoft 是 COM 和 OLE 的先祖，基于此种技术的应用软件，如 Microsoft Word, Excel, Access 得到广泛地使用和承认，而且许多应用软件也是基于这种技术，CAA 就是基于这种技术的 Windows 应用软件。



CAA 产品的架构可用上图表示。它全面反映了 Dassault Systemes 几大产品之间的关系。在 CAA 架构的支撑之下，Dassault Systemes 系统可像搭积木一样建立起来，这种结构非常适宜于系统的壮大和发展。下表列出了在 CATIA V5 应用方面为 Dassault Systemes 开发应用程序的部分公司。

序号	合作伙伴	产品
1	CADDAM Systems company Inc.	Helix Integration/Environment V1R4 CAA V5 Based/Helix Direct Interface (HDI)CAA V5 Based
2	Dimensional Control Systems Inc.	3DCS Analyst CAA V5 Based/3DCS Designer CAA V5 Based
3	Trace Parts	Trace Parts CAA V5 Based
4	T-Systems ITS GmbH	VAMOS CAA V5 Based
5	Metalsoft Inc.	Fabriwin CAA V5 Based
6	Mechanical Dynamics Inc.	Dynamic Designer CAA V5 Based
7	LMS International	LMS Virtual.Lab Acoustics LMS Virtual.Lab Noise and Vibration LMS Virtual.Lab Durability LMS Virtual.Lab Motion
8	ICEMCFD	ICEM CFD Hexa CAA V5 Based

9	<b>AIKOKU ALPHA ENGINEERING CORP.</b>	<b>AIKOKU Post for 4 axis CAA V5 Based/ AIKOKU Post for 5 axis CAA V5 Based</b>
10	<b>CENIT AG Systemhaus</b>	<b>CUT4AXES CAA V5 Based</b>
11	<b>EADS Matra Datavision</b>	<b>EUCLID3 Connectivity CAA V5 Based</b>
12	<b>ZEH Software</b>	<b>ZEHSever CAA V5 Based</b>
13	<b>Infrastructure Interface</b>	<b>COM/VDAFS CAA V5 Based</b>

在商业运作方面，与 Dassault Systemes 系统的其产品一样，CAA 也被划分为软件包（Configuration）和模块（Product），用户在购置 CAA 时要根据需求选择合适的配置和产品。

目前 CAA 有五个配置软件包：

- CAA—Multi-workspace Application Building(简称 ABC)  
CAA 应用的基本配置.
- CAA—C++ Extended Development(简称 CDC)  
提供与 C++共存的的开发环境,即对 C++开发环境的客户化,用户除可使用 C++开发环境的所有工具外,还可使用 CAA 提供的独特的开发工具.可用于 CATIA 和 ENOVIA 的二次开发.
- CAA—C++ Base Development(简称 CDV)  
与 CDC 类似,但规模小,针对中小市场,可有限地运用于 CATIA V5 和 ENOVIA LCA 的客户化工作.
- CAA—Java Base Development(简称 JDV)  
运行于 JAVA 开发环境,适用于 ENOVIA LCA 客户端的定制.
- CAA—Legacy Data Integration Development Configuration(简称 LDC)  
提供数据接口工具,可使用户继承原有的数据.

CAA 提供的产品(模块)包括:

- CAA—C++ API Documentation Generator(CDG).
- CAA—C++ Source Checker(CSC)
- CAA—CAA Data Model Customizer(DMC)
- CAA—Java UnitTest Manager(JUT)
- CAA—Multi-Worksoace Application Builder(MAB)
- CAA—Teamwork Release Manager(TRM)
- CAA—C++ Interactive Dashboard(CID)
- CAA—C++ Unit Test Manager(CUT)
- CAA—Java Interactive Dashboard(JID)
- CAA—Web Application Generator for Legacy Database(LWG)
- CAA—Source Code Manager(SCM)

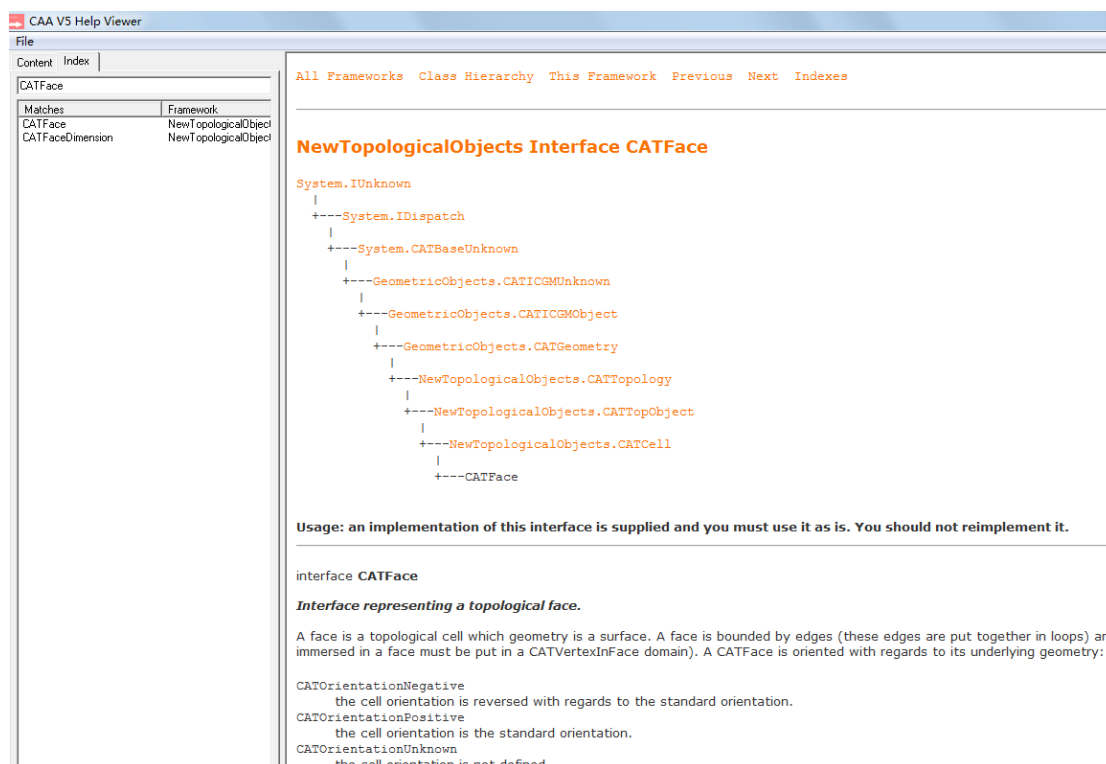
#### 1.1.4 帮助资料介绍

参考的帮助文档共三个：

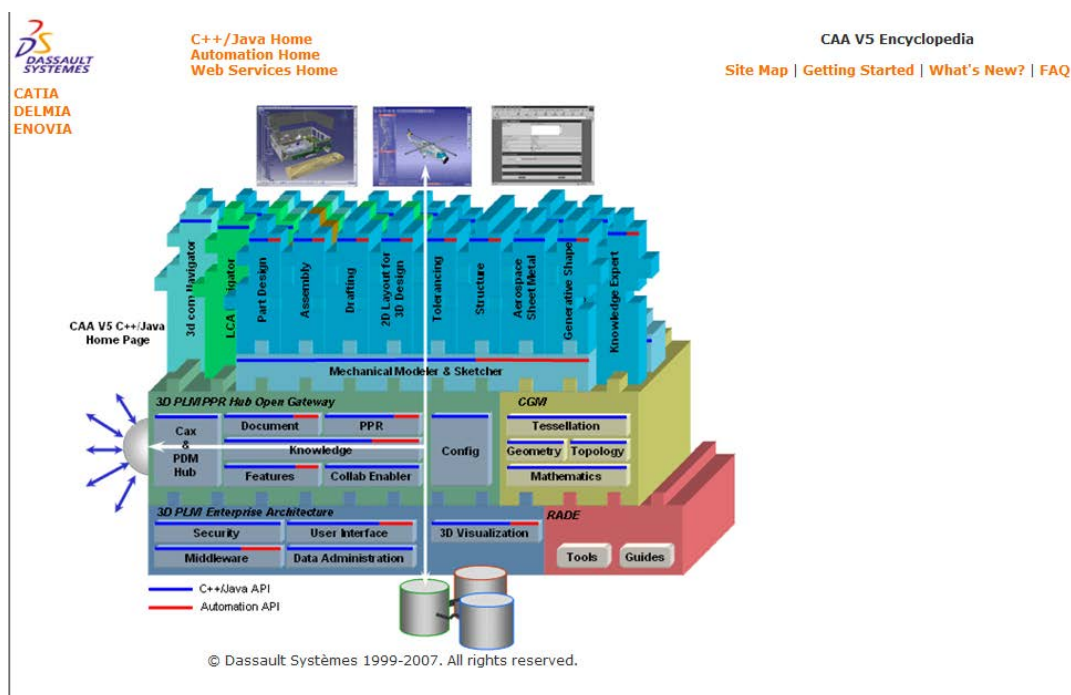
CNextHelpViewer.exe ("D:\Dassault Systemes\B19\intel\_a\code\bin\CNextHelpViewer.exe")

中提供了常用 Interface 及函数的定义及继承关系。

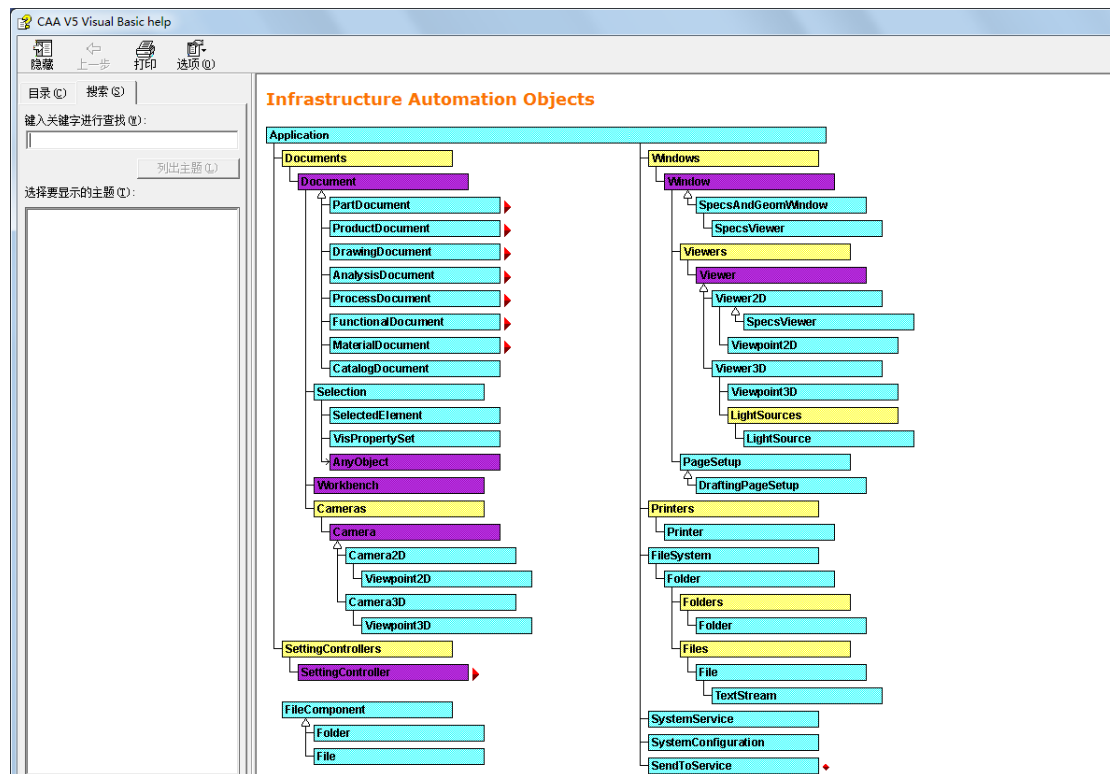




百科全书 CAAV5HomePage.htm ("D:\DassaultSystemes\B19\CAAV5HomePage.htm")提供了命名规则，编码规则，如何创建等实例和讲解。



CAA Automation 帮助文档：V5Automation.chm（路径为："D:\Dassault Systemes\B19\intel\_a\code\bin\V5Automation.chm"）



## 1.2 CATIA 二次开发架构

一个空间中至少一个 framework，一个 framework 中至少一个 module，所有实现的内容都是在 module 中完成的。

Workspace

The area where development projects are stored, equivalent to Solution in Visual Studio.

Framework

A framework collects a set of modules designed to work together to produce a certain “area” of functionality.

Module

A collect of program which is responsible for a certain functionality.

文档结构如下：

A Refined Structure of a CAA workspace

CAAXXXFrameWork（framework）

The must and basic part of the a workspace.

CAAXXXCmds（module）

The classes related to the logical processing of the functionality

CAAXXXAddins（module）

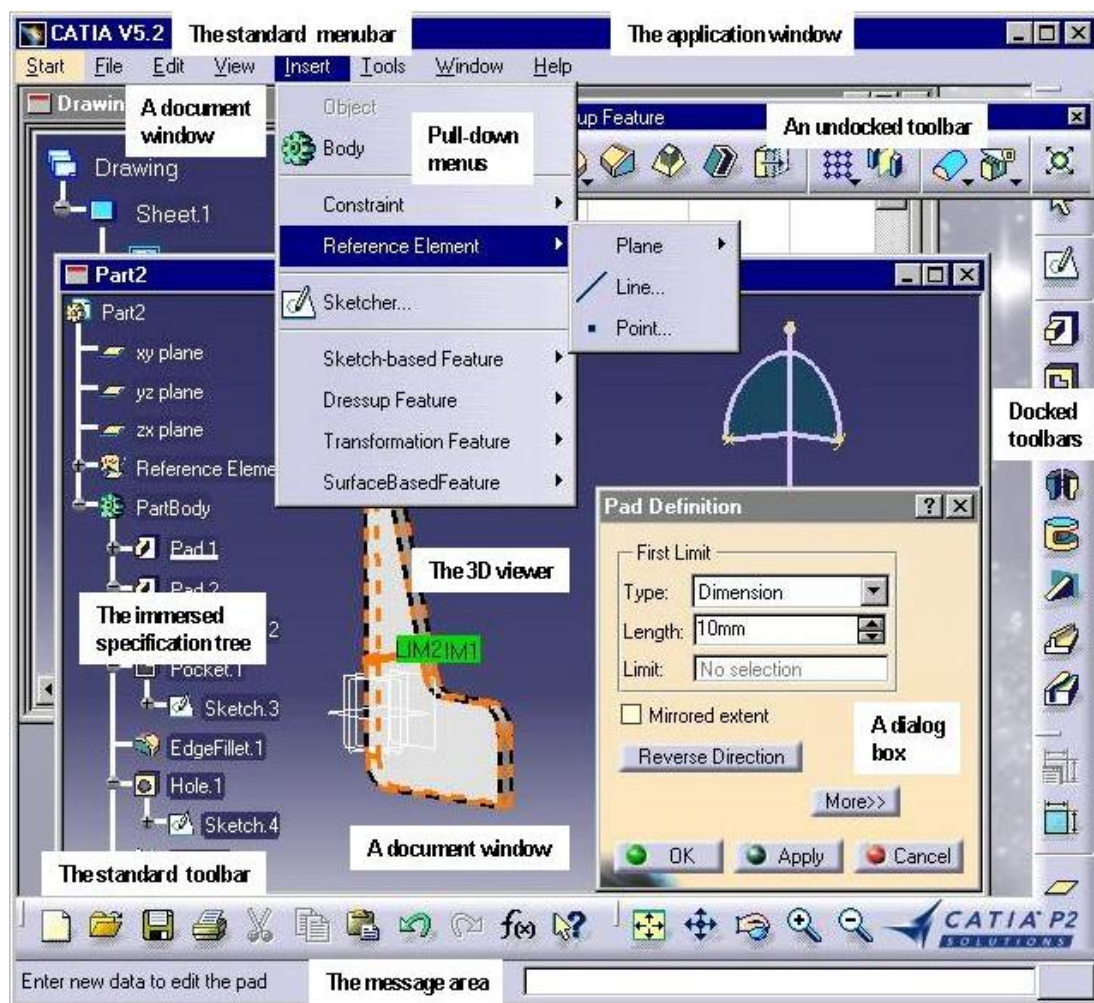
The Interface to the clients, Menus, toolbars, etc..

CAAForeignCmds (module)

Other classes imported from other frameworks, which may be responsible for certain special logical processing.

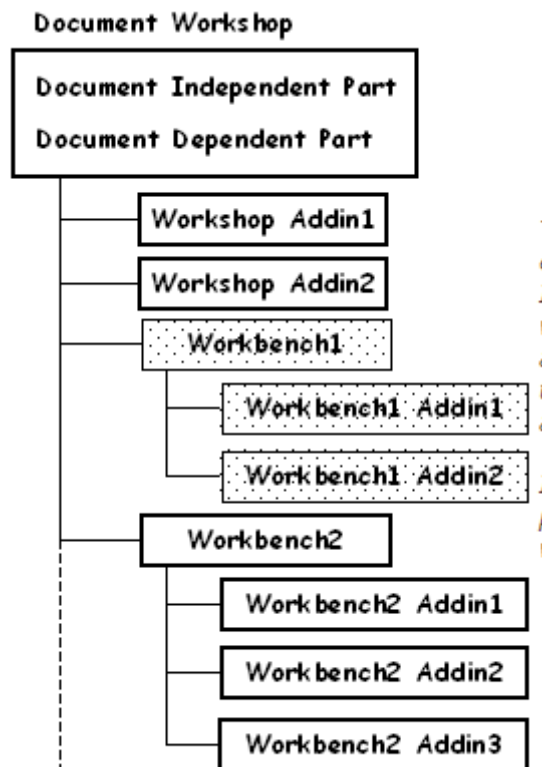
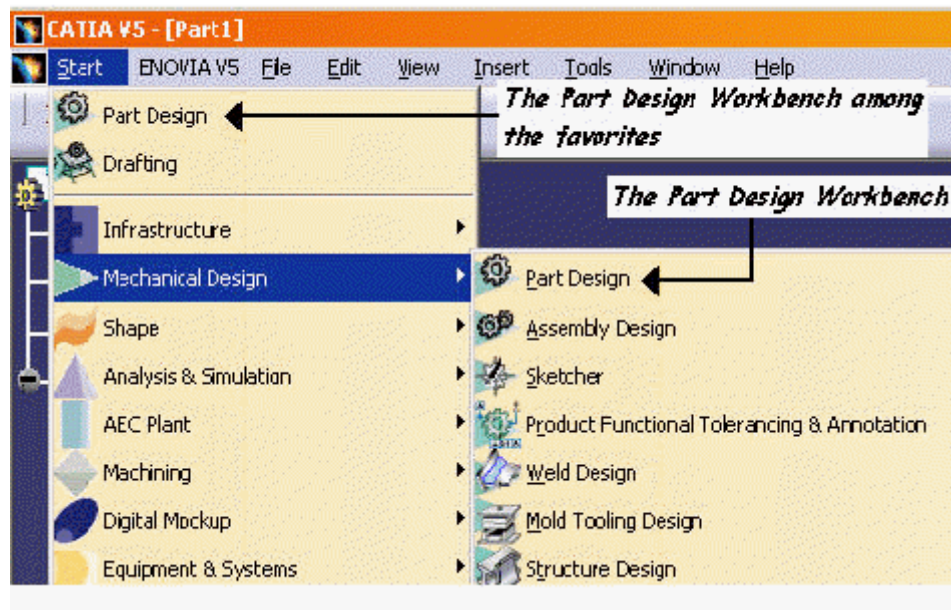
## 1.3 CATIA 主界面概述

主界面框架



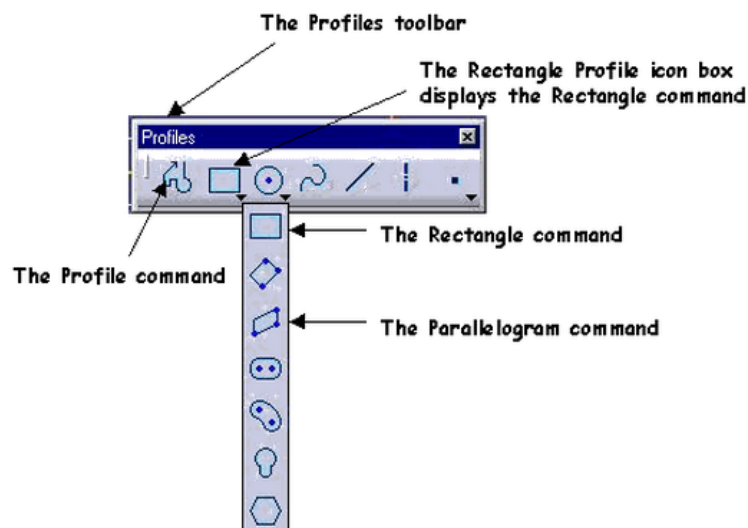
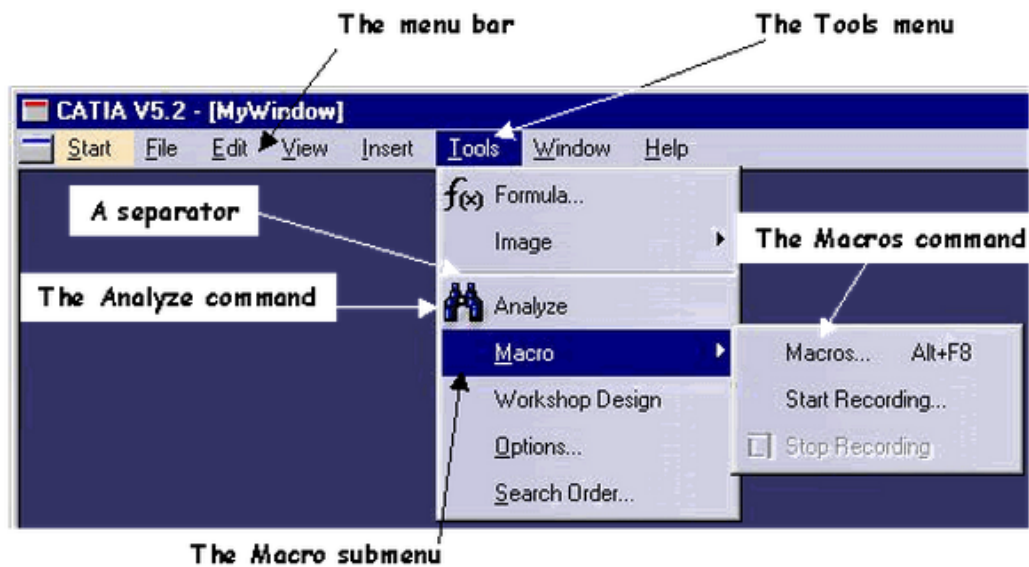
A CAA V5 document is associated with its own editing tools gathered in a workshop. Editing tools are commands that you arrange in menus and toolbars that make up the workshop.

Workshop 是由与 CATIA 文件独立的命令组成，workshop 由命令组成，包括如在“文件”菜单中的“新建”“打开”等这些命令无论何时都是可用的和必须激活某个 workbench 后才可以使用的，比如基于草图的特征。一个 workshop 可用包括很多个 workbench，一个 workbench 包括很多个 add-in。同时只有一个 workbench 可用。



一个 CATPart 文件打开后就是一个 workshop，默认进入的是 PartDesign 这个 workbench，在这个 workbench 中有几个 partdesign 特有的 addin 比如“基于草图的特征”子菜单和工具条。

For example, the Part document's workshop includes a Part Design workbench to create shapes, and a Free Style workbench to create curves and surfaces, and to modify and analyze shapes.



## 2 工程创建

### 2.1 添加工具条和菜单

#### 2.1.1 新建工作空间（workspace）和框架（frame）

文件——>new CAA V5workspace



## New CAA V5 Workspace

### New Workspace

Create a new Workspace where you will create new objects or edit existing ones.

With:

Workspace Directory:

...

Tool level:

Cancel

< Previous

Next >

## New CAA V5 Workspace

### New Workspace

Choose what you want to do first in your new workspace:

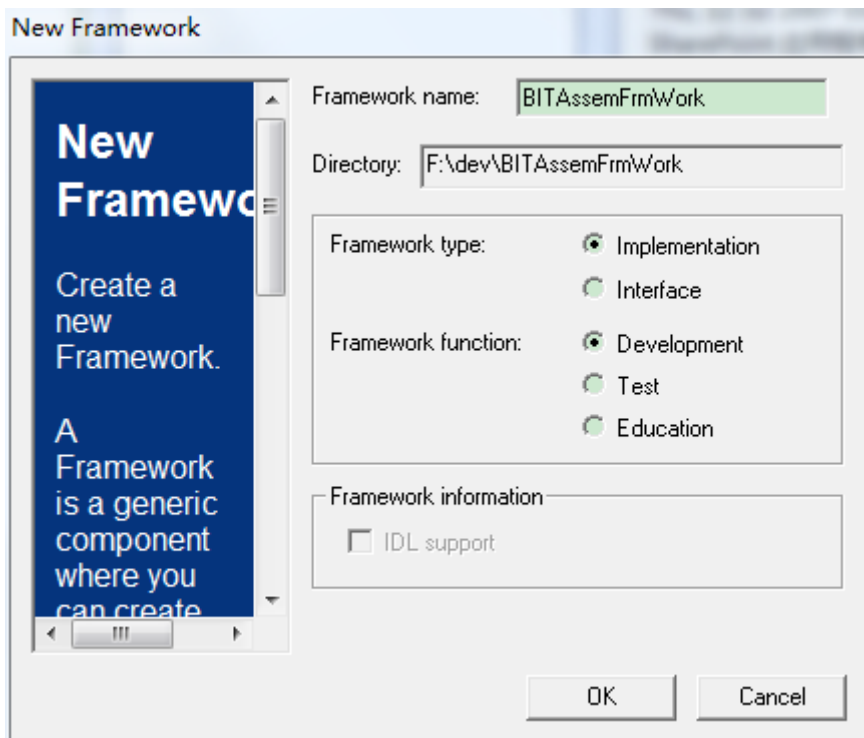
- If you have access to Workspace Manager tools, you can modify an existing framework

- ☐ Copy and modify frameworks through the WorkspaceManager
- ☐ Copy and modify frameworks from a reference workspace
- ☒ Create new generic framework
- ☐ Define Prerequisite Workspace
- ☐ Migrate a workspace

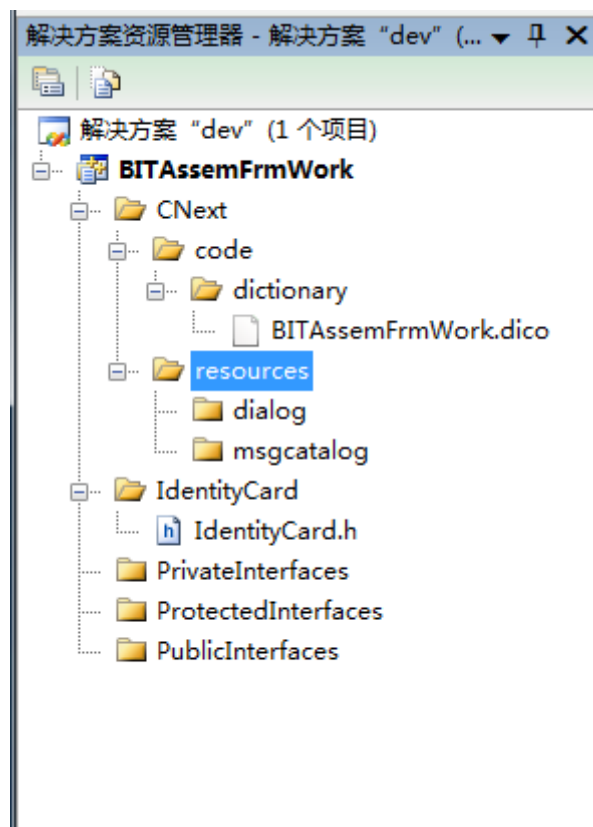
Cancel

< Previous

Finish



文档结构



修改 identitycard 中 identitycard.h 文件，这个文件中的内容为当前 frmwork 中可能用到的接口，需要在此加载。粘贴

```
AddPrereqComponent("ApplicationFrame", Protected);  
AddPrereqComponent("CATAssemblyInterfaces", Protected);
```

---

```
AddPrereqComponent("ObjectModelerBase", Protected);
AddPrereqComponent("ProductStructureUI", Protected);
AddPrereqComponent("DialogEngine", Protected);
AddPrereqComponent("Mathematics", Protected);
AddPrereqComponent("Dialog", Protected);
AddPrereqComponent("MechanicalModelerUI", Protected);
AddPrereqComponent("MechanicalModelerUI", Public);
AddPrereqComponent("PartInterfaces", Public);
AddPrereqComponent("ObjectSpecsModeler", Public);
AddPrereqComponent("ConstraintModeler", Public);
AddPrereqComponent("ConstraintModelerInterfaces", Public);
AddPrereqComponent("ConstraintModelerUI", Public);
AddPrereqComponent("SketcherInterfaces", Public);
AddPrereqComponent("GeometricOperators", Public);
AddPrereqComponent("ProductStructureInterfaces", Public);
AddPrereqComponent("ProductStructure", Public);
AddPrereqComponent("SpaceAnalysisInterfaces", Public);
AddPrereqComponent("SimulationBase", Public);
AddPrereqComponent("SimulationInterfaces", Public);
AddPrereqComponent("GSMInterfaces", Public);
AddPrereqComponent("GSOInterfaces", Public);
AddPrereqComponent("LiteralFeatures", Public);
AddPrereqComponent("DraftingInterfaces", Public);
AddPrereqComponent("CATTTTRSInterfaces", Public);
AddPrereqComponent("CATTPSInterfaces", Public);
AddPrereqComponent("ManufacturingInterfaces", Public);
AddPrereqComponent("SurfaceMachiningInterfaces", Public);
AddPrereqComponent("DMAPSInterfaces", Public);
AddPrereqComponent("GeometricObjects", Public);
AddPrereqComponent("KnowledgeInterfaces", Public);
AddPrereqComponent("LiteralsEditor", Public);
AddPrereqComponent("MechanicalModeler", Public);
AddPrereqComponent("Tessellation", Public);
AddPrereqComponent("Visualization", Public);
AddPrereqComponent("VisualizationBase", Public);
AddPrereqComponent("MecModInterfaces", Public);
AddPrereqComponent("NewTopologicalObjects", Public);
AddPrereqComponent("MechanicalCommands", Public);
AddPrereqComponent("TopologicalOperators", Public);
AddPrereqComponent("InteractiveInterfaces", Public);
AddPrereqComponent("PrismaticMachiningInterfaces", Public);
AddPrereqComponent("ToolPathEditorInterfaces", Public);
AddPrereqComponent("FreeFormOperators", Public);
AddPrereqComponent("CATIAApplicationFrame", Public);
```

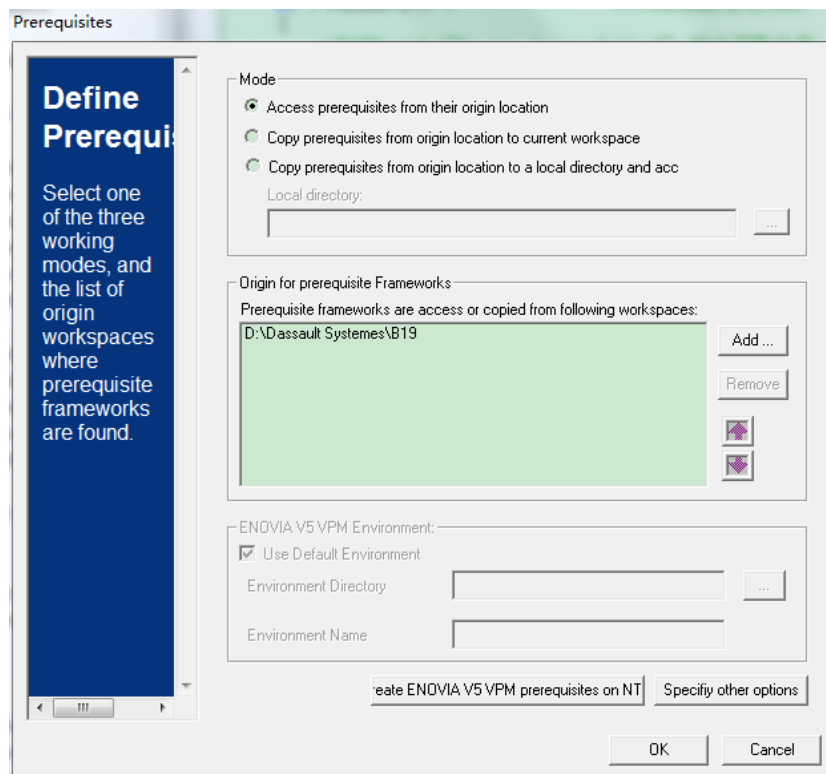


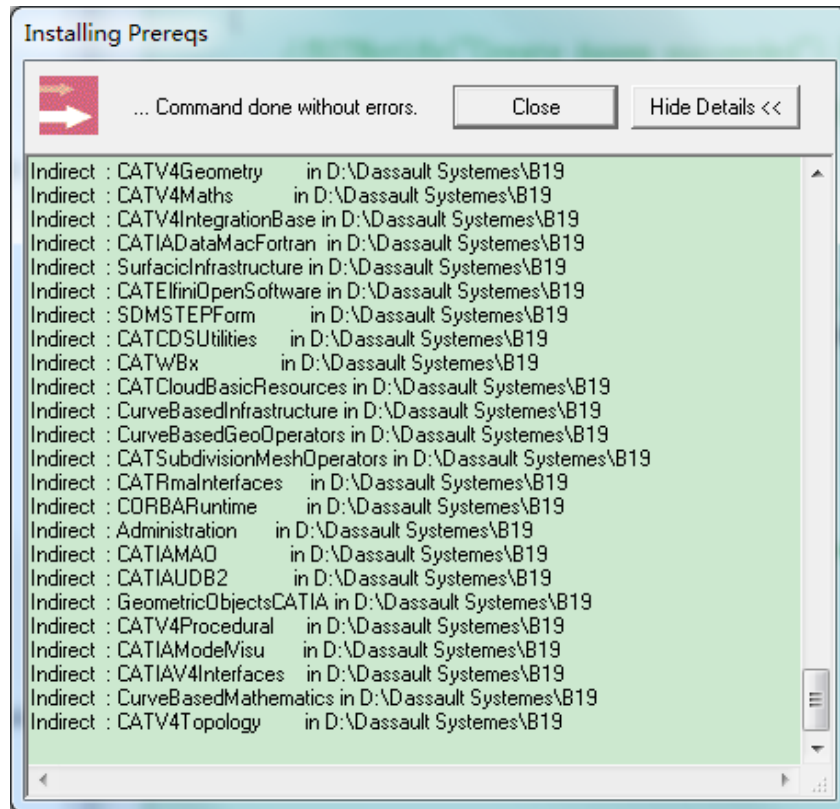
```
AddPrereqComponent("ComponentsCatalogsInterfaces", Public);
AddPrereqComponent("ProcessPlatformBase", Public);
AddPrereqComponent("ProcessPlatformVisu", Public);
AddPrereqComponent("AdvancedMachiningInterfaces", Public);
AddPrereqComponent("ObjectModelerInterfaces", Public);
AddPrereqComponent("SketcherModeler", Public);
AddPrereqComponent("MeasureGeometryInterfaces", Public);
AddPrereqComponent("InfInterfaces", Public);
AddPrereqComponent("XMLParser", Public);
AddPrereqComponent("AdvancedMathematics", Public);
AddPrereqComponent("CATPlantShipModeler", Public);
AddPrereqComponent("CATSchPlatformInterfaces", Public);
```

这些接口基本上包含了在编程过程中用到的所有接口，一次性粘贴一劳永逸。

完了之后 locat prerequisite workspace。

CAAV5 Workspace——locat prerequisite workspaces，选择到 RADE 安装 B19 文件夹

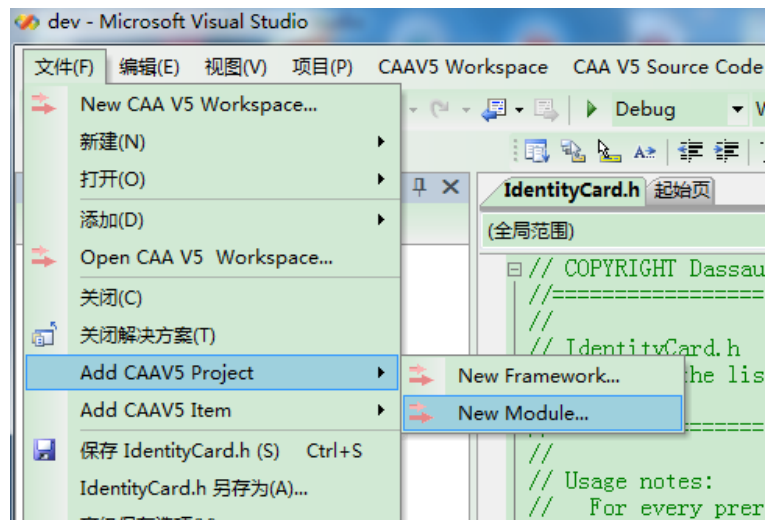


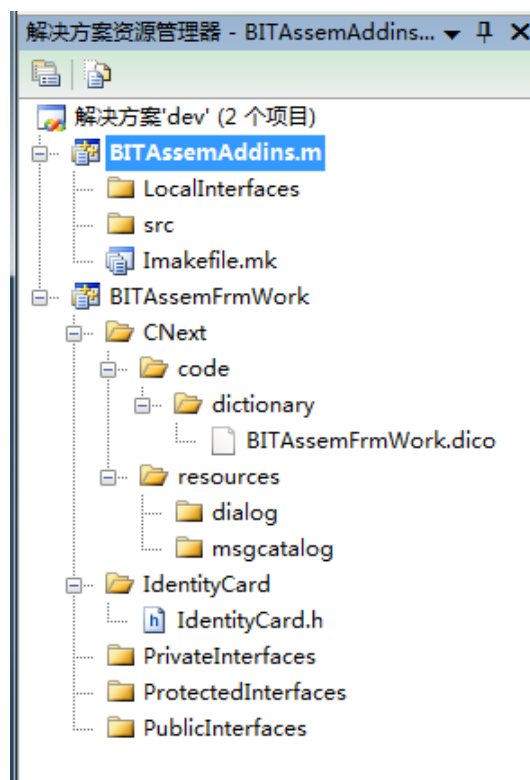
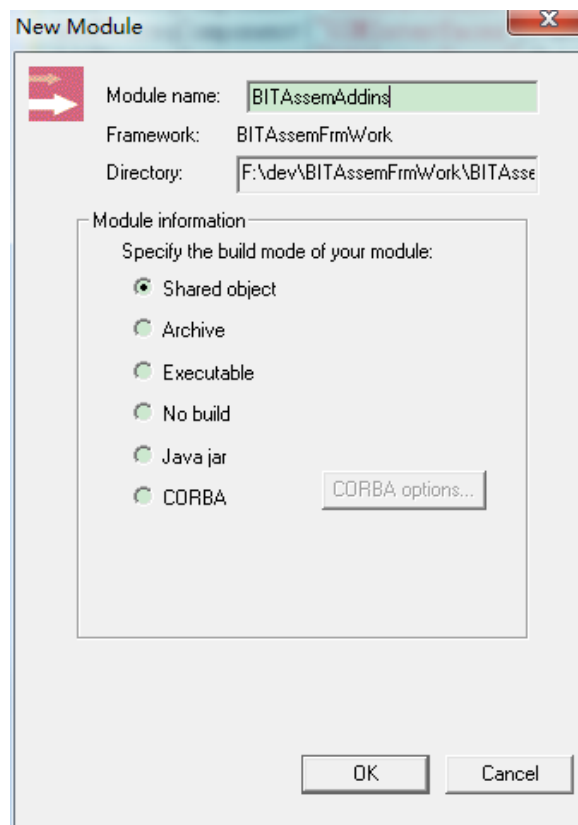


无错误后点击 close 关闭

## 2.1.2 添加模块（Module）和组件（component）

添加 addin module: BITAssemAddin.m 存放用户界面等。





### 添加组件 (component)

新建名称为 BITAssemAddin 的 Component。

BITAssemAddin 是用来添加菜单和工具条代码的模块。

**Insert Component**

Component name: BITAssemAddin

Derived from: CATBaseUnknown/C ...

Interfaces adhesion

BOA mode

TIE mode

TIEchain mode

☐ Check interfaces existence in prerequisite workspaces (slower)

File information

Framework: BITAssemFrmWork

Module project: BITAssemAddins.m

Header repository: LocalInterfaces

OK Cancel

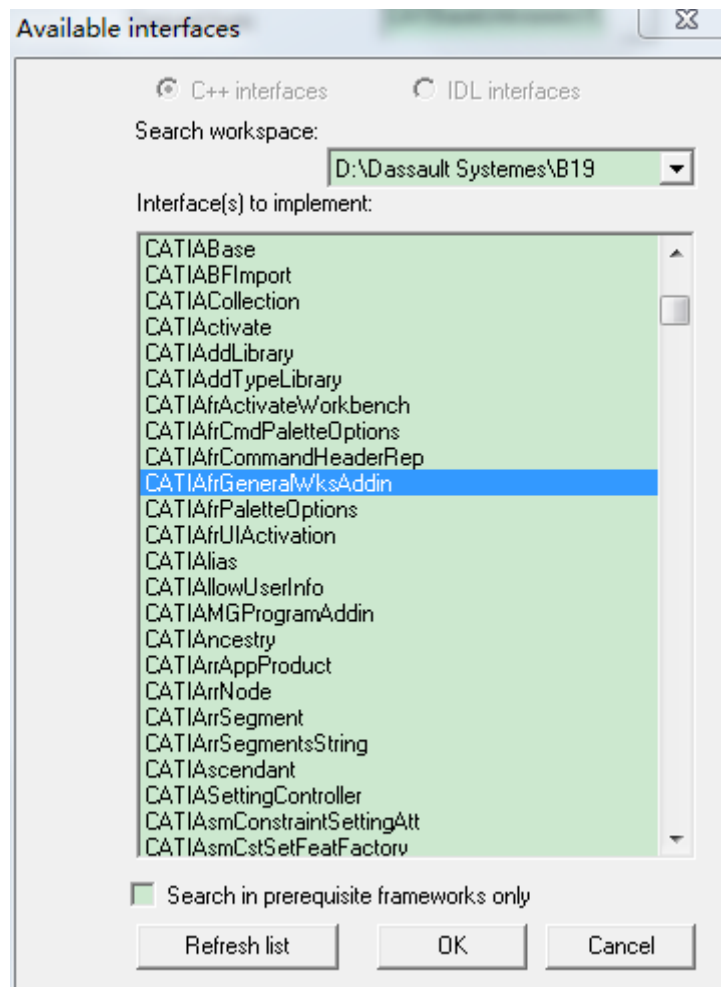
TIE mode 选择:

点击 add

**Interface :**

OK Cancel

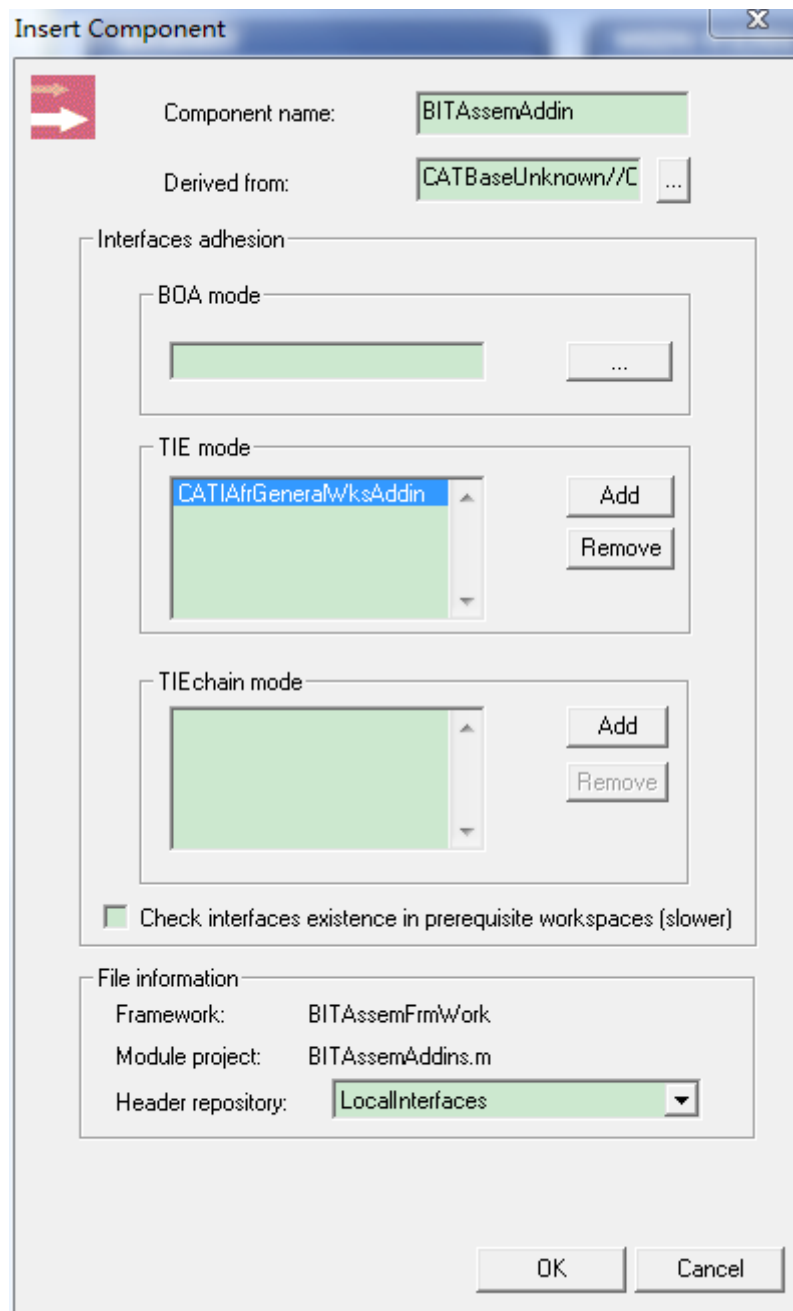
点击...



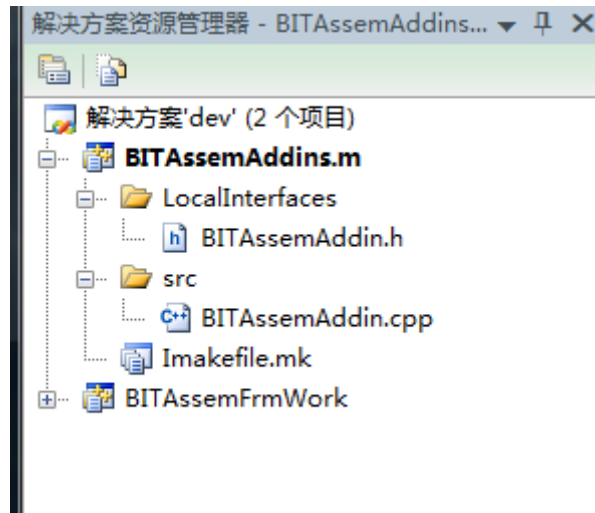
Search workspace 选择 B19 目录

取消勾选下面 search in prerequisite frameworks only

选择 CATIAfrGernalWksAddin

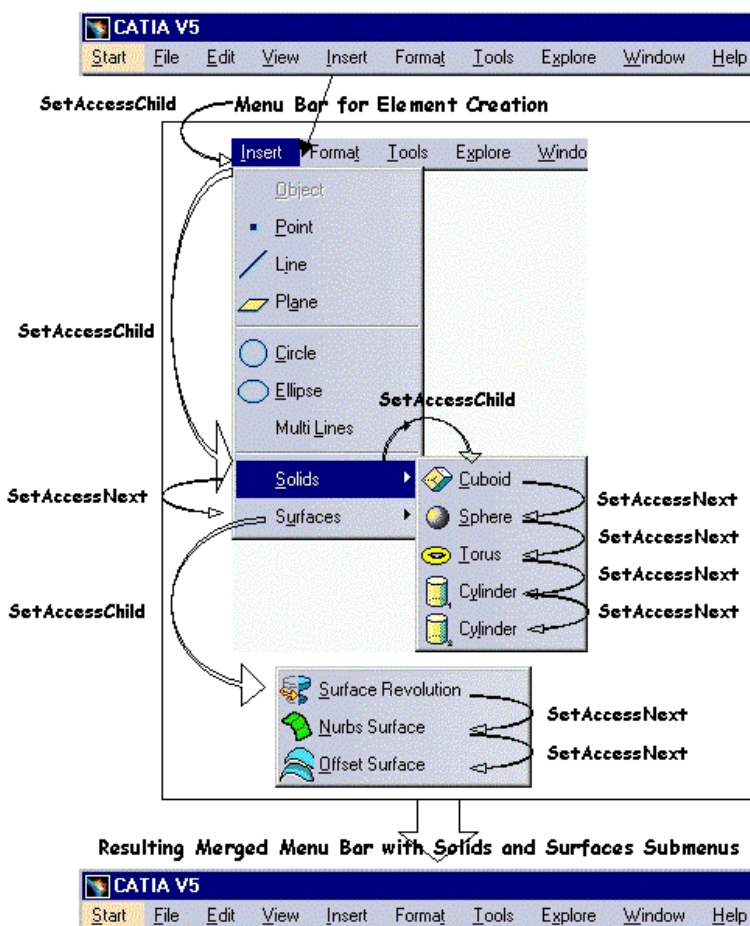


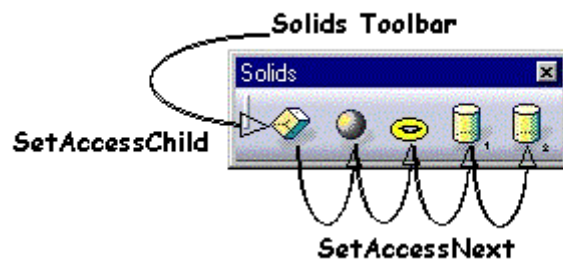
点击 OK 确定。文档结构树中新增



在 Imakefile 中 WIZARD\_LINK\_MODULES= 后添加 JS0FM JS0GROUP ApplicationFrameCATAfrUUID。这些是 module 用到的模块。

### 2.1.3 添加相应代码





在 BITAssemAddin.h 中添加头文件

```
#include "CATCmdContainer.h" //创建container时用到
#include "CATCommandHeader.h" //创建header时用到
#include "CATCreateWorkshop.h"
```

在 BITAssemAddin 构造函数中添加两个函数

```
void CreateCommands() ; //创建命令
CATCmdContainer *CreateToolbars() ; //创建工具条
```

在 BITAssemAddin.cpp 中添加

宏定义命令头文件: MacDeclareHeader(BITAssemHeader);宏定义 BITAssemHeader 头文件。

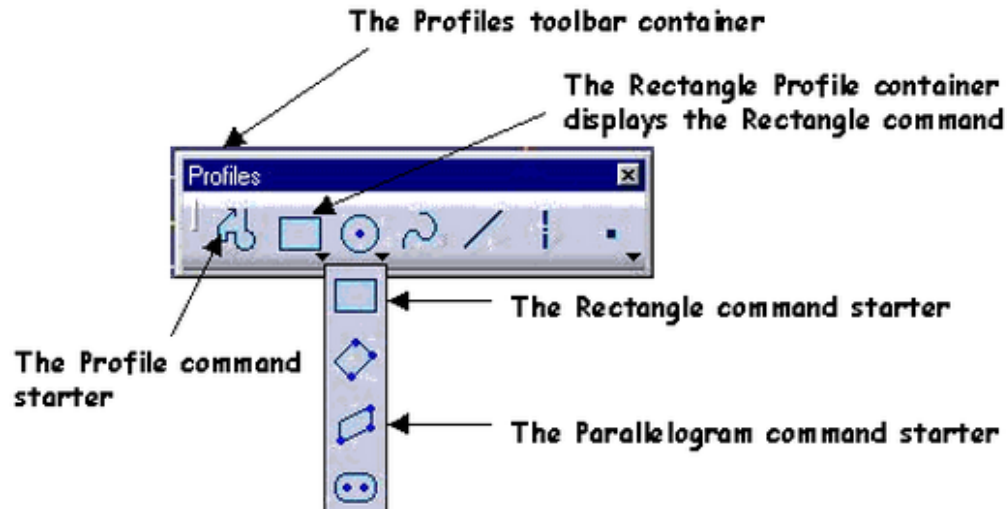
编写 void BITAssemAddin::CreateCommands()的实现

```
void BITAssemAddin::CreateCommands()
{
    //此语句用于将command与工具条联系起来
    //参数: 菜单名
    //参数: 菜单相应的命令所在的模块名
    //参数: 插入的命令名
    new BITAssemHeader("BITAssemHdr", "BITAssemCmds", "BITCreateAssemCmd",
(void*)NULL);
    new BITAssemHeader("BITAssemSndHdr", "BITAssemCmds", "BITCreateAssemSndCmd",
(void*)NULL);
}
```

编写 CATCmdContainer \* BITAssemAddin::CreateToolbars()的实现

新建工具条容器, 新建工具命令 1, 新建工具命令 2, 新建菜单



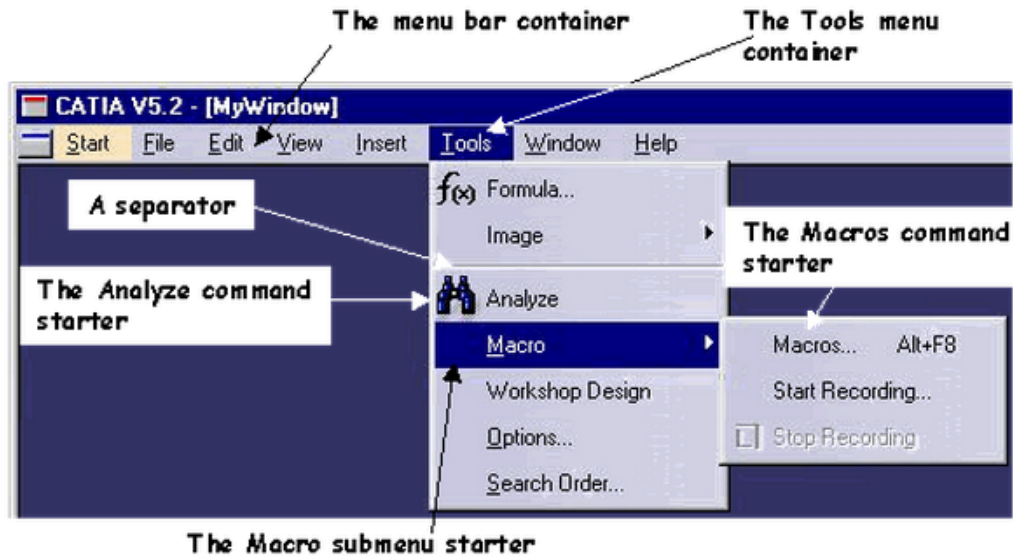


```
CATCmdContainer * BITAssemAddin::CreateToolbars()
{
    //1. 添加工具条
    NewAccess(CATCmdContainer, pCAAPMST1b, CAAPMST1b);

    //2. 1. 添加按钮.
    NewAccess(CATCmdStarter, pCAATPMSCreateLineStr, CAATPMSCreateLineStr);
    //2. 2. 给按钮关联命令
    SetAccessCommand(pCAATPMSCreateLineStr, "BITAssemHdr");
    //2. 3. 将按钮添加至工具条中
    SetAccessChild(pCAAPMST1b, pCAATPMSCreateLineStr);

    //3. 1. 添加按钮
    NewAccess(CATCmdStarter, pCAATPMSCreateLineSndStr,
CAATPMSCreateLineSndStr);
    //3. 2. 给按钮关联命令
    SetAccessCommand(pCAATPMSCreateLineSndStr, "BITAssemSndHdr");
    //3. 3. 将按钮添加至工具条中
    SetAccessNext(pCAATPMSCreateLineStr, pCAATPMSCreateLineSndStr);

    //4. 将工具条添加到视窗中
    AddToolbarView(pCAAPMST1b, 1, Right);
}
```



//1. 添加菜单

```
NewAccess(CATCmdContainer, pCAAPMSMnu, CAAPMSMnu);
```

//添加菜单窗口

```
NewAccess(CATCmdContainer, pCATAssemMnu, CATAssemMnu);
```

```
SetAccessChild(pCAAPMSMnu, pCATAssemMnu);
```

//2. 添加菜单项

```
NewAccess(CATCmdStarter, pCAATPMuAixsAutoStr, CAATPMuAixsAutoStr);
```

```
SetAccessCommand(pCAATPMuAixsAutoStr, "BITAssemHdr");
```

```
SetAccessChild(pCATAssemMnu, pCAATPMuAixsAutoStr);
```

//3. 添加菜单项

```
NewAccess(CATCmdStarter, pCAATPMuAixsAutoSndStr, CAATPMuAixsAutoSndStr);
```

```
SetAccessCommand(pCAATPMuAixsAutoSndStr, "BITAssemSndHdr");
```

```
SetAccessNext(pCAATPMuAixsAutoStr, pCAATPMuAixsAutoSndStr);
```

//4. 将菜单和工具条关联

```
SetAddinMenu(pCAAPMST1b, pCAAPMSMnu);
```

//Attach the menu bar to the workbench

```
return pCAAPMST1b;
```

```
}
```

NewAccess(新建 header, container, starter 等)

SetAccessCommand (给对相关命令)

SetAccessChild (设置子级关系)

SetAccessNext (设置并列关系)

AddToolbarView (添加工具条显示)

SetAddinMenu

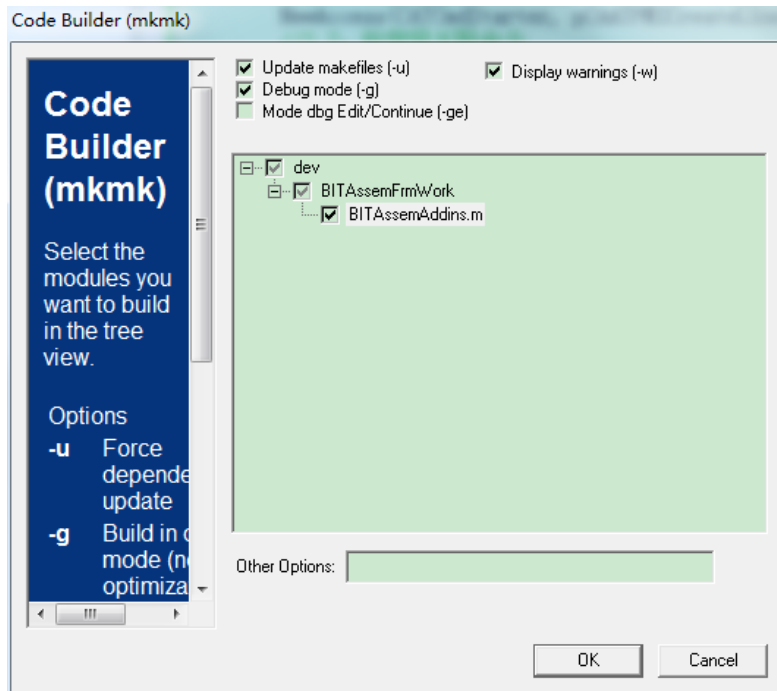
CATCmdSeparator 分割线

CATCmdStarter

运行程序：

CAAV5 Workspace->Create/Update Runtime View->Ok

生成->mkmk->OK



窗口->Open runtime Window-输入”Cnext”->回车

## 2.1.4 CATNls 和 CATRsc 的使用

定义 header 资源文件，名称图标等。Name 为程序中的引用时的代号。Title 为用户使用时界面上显示的名称。Creat/update runtime view 整合加载的模块的资源。改了界面相关的东西时运行下，其他时候不需要。

在 BITAssemFrmWork\CNext\resources\msgcatalog 中新建两个文档，先新建文本文档然后修改后缀名，BITAssemAddin.CATNls（存放显示的名称）BITAssemAddin.CATRsc（存放图片等资源文件）和 BITAssemHeader.CATNls、BITAssemHeader.CATRsc。

BITAssemAddin.CATNls 中写入：

```
CAAPMSTlb.Title = “装配”;  
CAAPMSMnu.Title = “装配”;  
CATAssemMnu.Title = “装配”;
```

BITAssemAddin.CATRsc 暂不写入。

BITAssemHeader.CATNls 中写入：

```
BITAssemHeader.BITAssemHdr.Title = “装配”;
```

```
BITAssemHeader.BITAssemHdr.ShortHelp = "装配";  
BITAssemHeader.BITAssemHdr.Help = "装配";
```

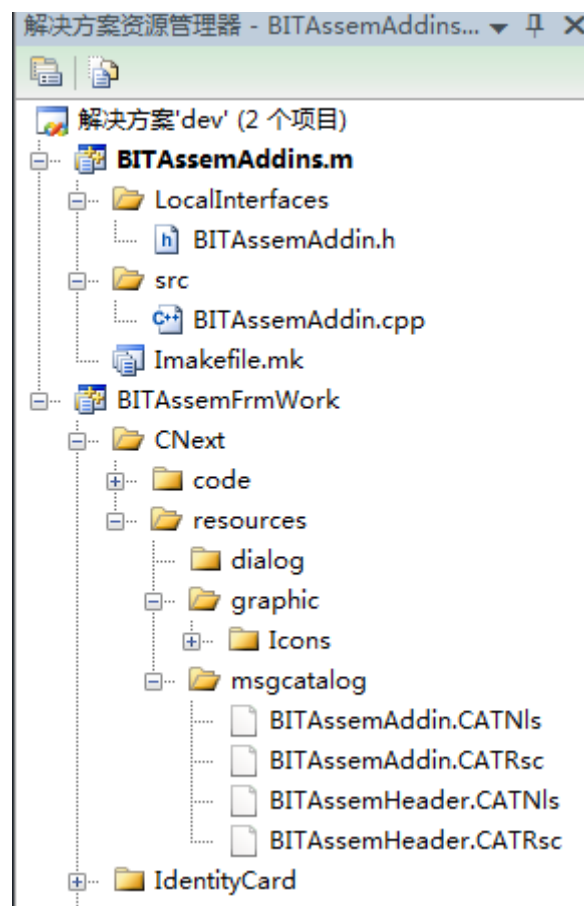
```
BITAssemHeader.BITAssemSndHdr.Title = "装配";  
BITAssemHeader.BITAssemSndHdr.ShortHelp = "装配";  
BITAssemHeader.BITAssemSndHdr.Help = "装配";
```

```
BITAssemHeader.CATRsc 中写入  
BITAssemHeader.BITAssemHdr.Icon.Normal = "I_Skeleton";  
BITAssemHeader.BITAssemHdr.LongHelpId = "BITAssemHeader.BITAssemHdr";
```

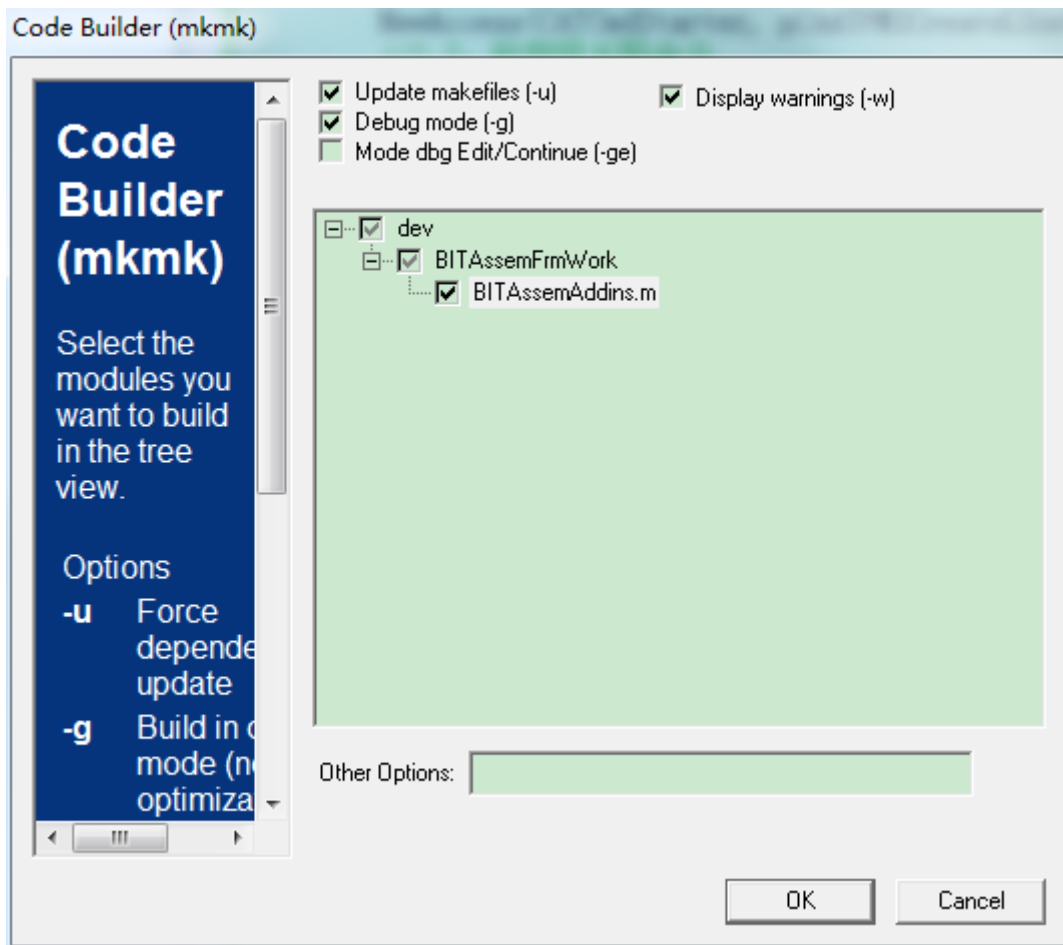
```
BITAssemHeader.BITAssemSndHdr.Icon.Normal = "I_CreateLine";  
BITAssemHeader.BITAssemSndHdr.LongHelpId = "BITAssemHeader.BITAssemSndHdr";
```

将例程中 BITAssemFrmWork\CNext\resources\graphic 文件夹整个拷贝到自己的 BITAssemFrmWork\CNext\resources 中。然后刷新工程。

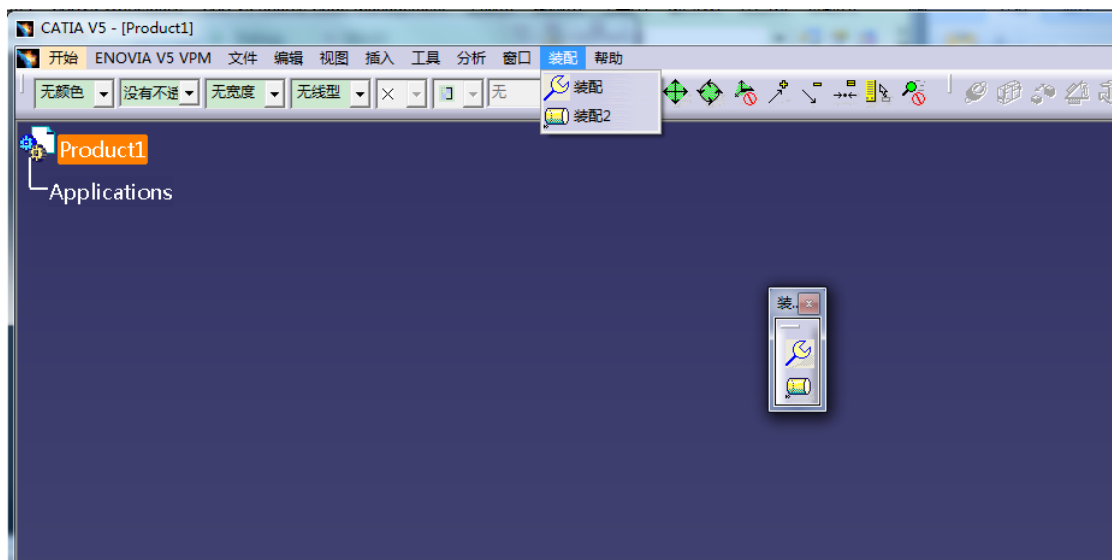
文档结构如下：



然后 Create/update runtime view 整合加载的模块的资源,照之前的做法生成 mkmk 并运行。



如下图，界面中出现了菜单和工具条。



---

## 2.2 新建命令（Command）和对话框（Dialog）

### 2.2.1 创建新的模块（module）

接下来要为命令创建响应。弹出对话框

设置 BITAssemFrmWork 为当前启动项。添加 module: BITAssemCmds.m。名称与之前 new BITAssemHeader("BITAssemHdr", "BITAssemCmds", "BITCreateAssemCmd", (void\*)NULL);中第二个参数相同，参见注释。修改其 imakefile.m 文件，在 WIZARD\_LINK\_MODULES =后添加

```
DIOPANV2 CATMathematics CATDialogEngine \  
    CATSchItfCPP CATDraftingInterfaces CATTPSItf \  
        JSOGROUP JSOFM CATPspUtilities \  
            CATApplicationFrame CATViz \  
                CATMecModInterfaces CATObjectSpecsModeler \  
                    CATGeometricObjects CATMathStream CATTopologicalObjects \  
                        CATGeometricOperators CATObjectModelerBase KnowledgeItf \  
                            CATGitInterfaces CATSketcherInterfaces CATVisualization \  
                                CATMeasureGeometryInterfaces CATInteractiveInterfaces \  
                                    CATProductStructure1 CATProcessInterfaces CATPartInterfaces \  
                                        CATTopologicalOperators CATManufacturingInterfaces  
CATMechanicalModeler \  
    JSOFM JSOGROUP CATXMLParserItf \  
        ADOXXBAS CDOFRAME \  
        DIOSTATE ACOSPBAS \  
        ProcessInterfaces \  
        CATManufacturingInterfaces \  
        CATSurfaceMachiningInterfaces \  
        CATProductStructure1 \  
            CATLiteralFeatures KnowledgeItf \  
            CATIAApplicationFrame \  
            CATProductStructure1 \  
            KnowledgeItf \  
            CATPartInterfaces CATSketcherInterfaces \  
            CATMechanicalModeler \  
            CATGitInterfaces \  
            CATVisualization \  
            CATAssemblyInterfaces \  
            CATConstraintModeler \  
            CATMathStream \  
            CATConstraintModelerItf \  
            CATUdfInterfaces \  
            JSOSCBAS \  
            JSO
```

---

```
CATTopologicalOperators \
CATInteractiveInterfaces \
Mathematics \
  CATPrismaticMachiningInterfaces CATPartInterfaces \
PrismaticMachiningInterfacesUUID \
CATSketcherInterfaces CATUdfInterfaces \
CATSaiSpaceAnalysisItf \
  CATMechanicalCommands \
  CATVisualization \
  CATCclInterfaces \
  CATInteractiveInterfaces \
    CATMeasureGeometryInterfaces ACOXXLNK \
    SpecsModeler\
    CKOFEAT JSOCORBA \
    YP00IMPL YN000MAT\
    CATCGMGeoMath \
      CATWkAssyInterface CATAssemblyInterfacesUUID \
    CATMechanicalModelerUI \
    CATTopologicalObjects \
    CATConstraintModeler \
    CATGeometricOperators \
    InfItf \
    InfItfCPP \
    CATTTRSItf \
    CATTTRSUUID \
    InfProIDL \
    GeoNurbsTools \
    CATFreeFormOperators \
    CATAdvancedMachiningInterfaces \
    ProductStructurePubIDL \
    CATTessellation CATAdvancedMathematics \
    CATTPSItfCPP \
    CATJNIAnnotationTypeLib \
    CATTPSProIDL \
    CATTPSPubIDL \
    CATTPSUUID \
    JSOGROUP JSOFM \
    CATPartInterfaces \
    CATSketcherInterfaces \
    CATVisualization \
  CATGeometricOperators \
CATProductStructure1 \
CATMathStream \
CATSaiSpaceAnalysisItf \
```

---

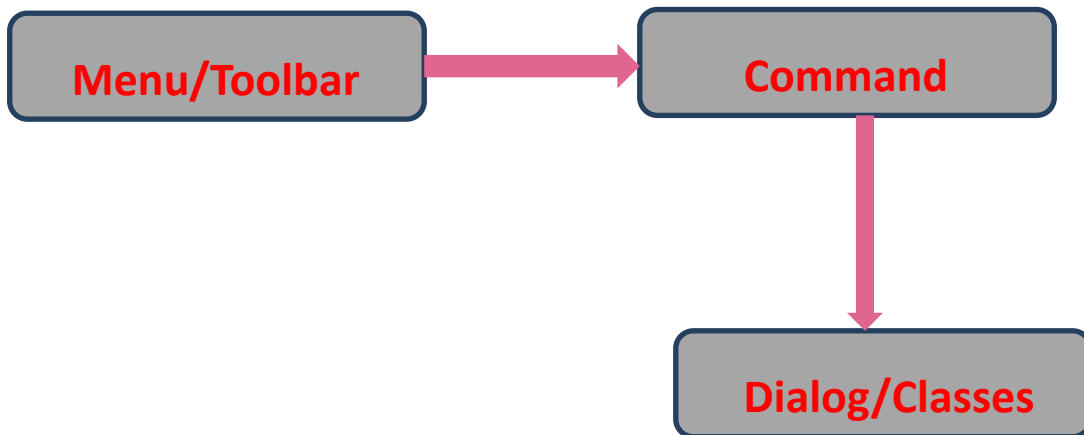
```

JS0SCBAK \
SimulationItf \
CATMechanicalModelerUI \
CATMechanicalModeler \
  CATProductStructure1 \
  CATTTRSItf \
  CATTTRSUUID \
  CATInteractiveInterfaces \
  CATTPSUUID \
  CATMathStream \
  CATCGMGeoMath \
  CATTopologicalObjects \
DIOPANV2 \
CATMathematics CATDialogEngine

```

以上基本包含了编程过程中用到的模块，跟之前 `identity.h` 中引用的接口一样，都是直接使用的，多了也没关系。这些内容是本模块在运行时需要调用内容或者引用的模块，如果缺少部分的话会导致出现无法解析的符号之类的错误。

命令响应顺序: 点击按钮链接到 `addin` 中 `header`—`cmd: active` 时的方法—调用 `dlg`.

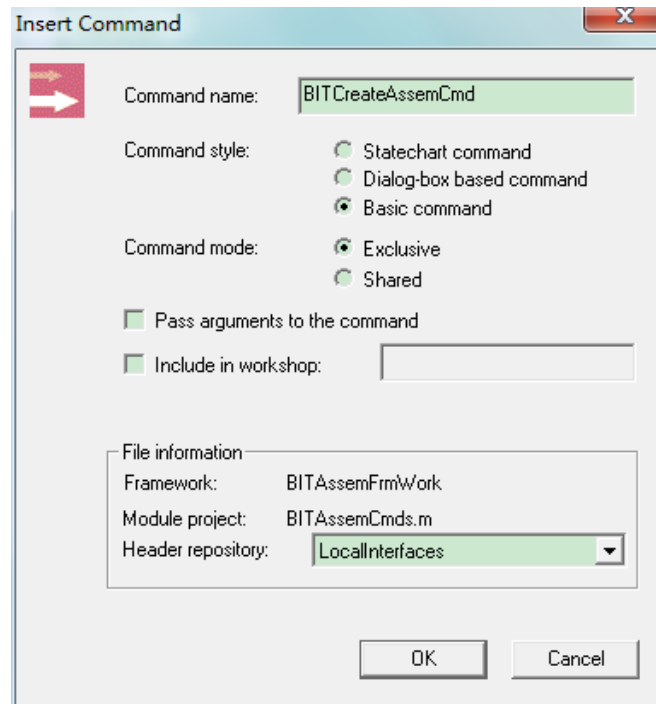


### 2.2.2 插入命令 (command)

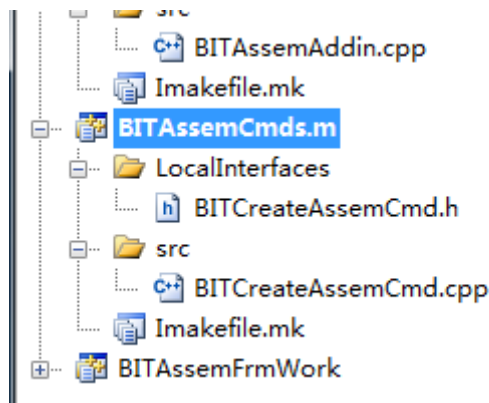
添加 command: `BITCreateAssemCmd`。

创建 command: `BITCreateAssemCmd` (文件——Add CAA V5 item——catia resource——command)。名称与之前 `new BITAssemHeader("BITAssemHdr", "BITAssemCmds", "BITCreateAssemCmd", (void*)NULL);` 中第三个参数相同，参见注释。三种类型，one-shot(basic), state chart, dialog-box。每种都有三种模式 active, desactive, cancle，具体参加见百科全书。此处选择 basic 模式，单击按钮时弹出对话框即可。





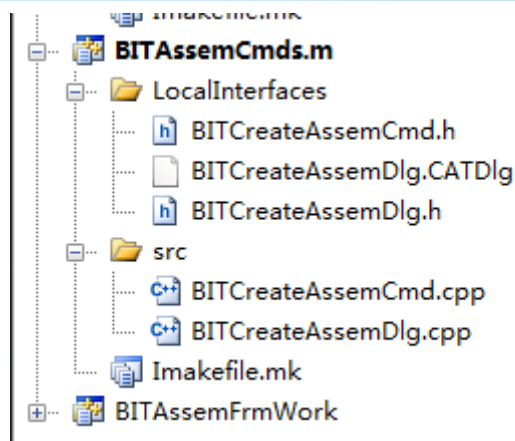
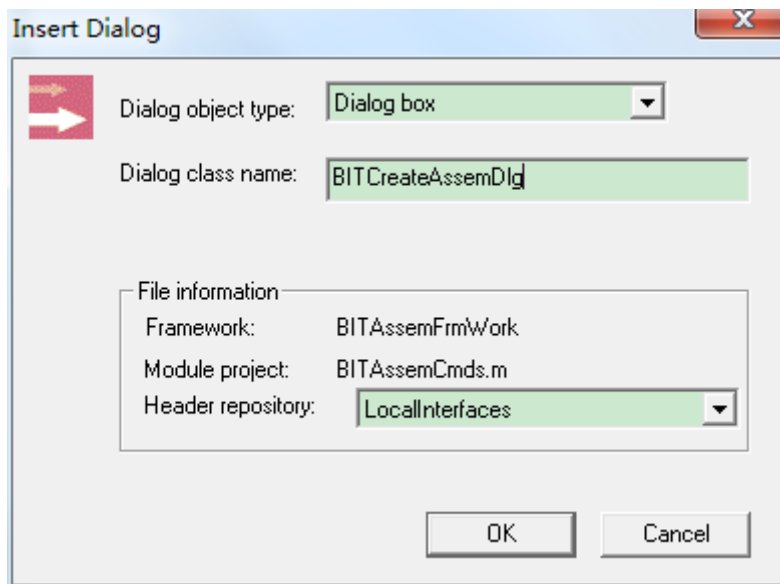
文件树中新生成两个文件：



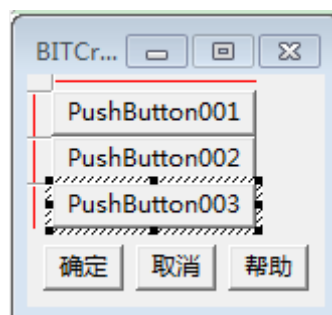
在 BITCreateAssemCmd.cpp 中添加头文件#include "BITCreateAssemDlg.h"

### 2.2.3 加入新的对话框（Dialog）

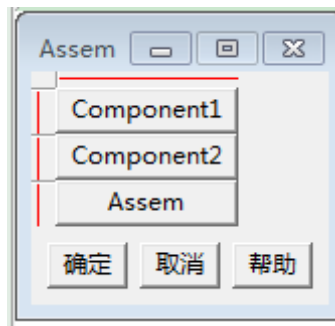
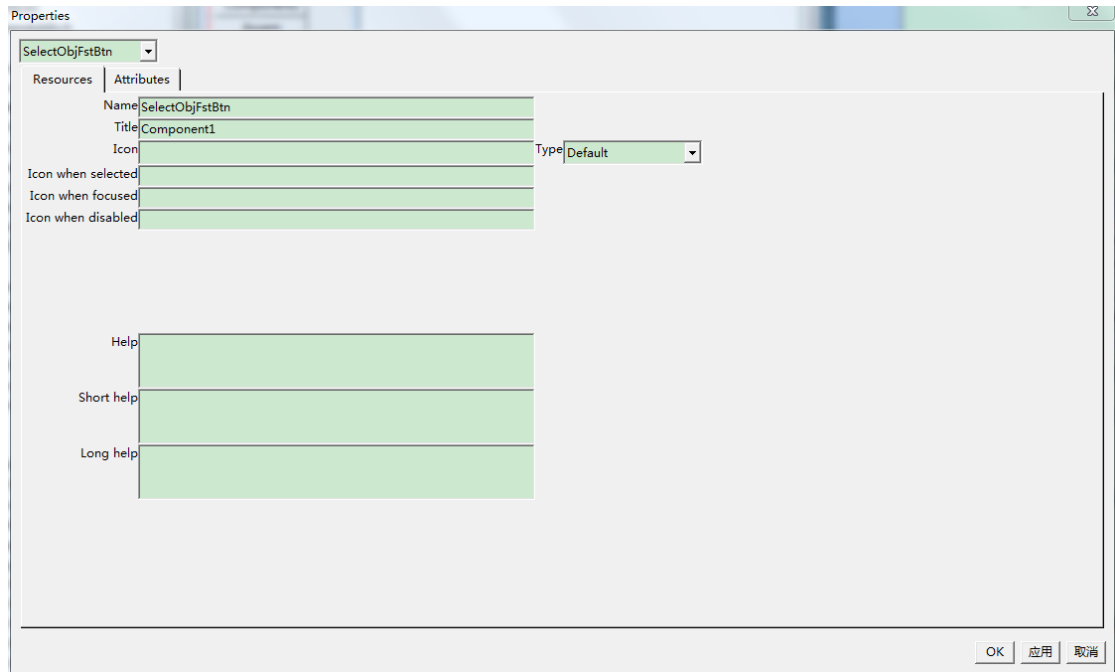
在 BITAssemCmds.m 里面新建 Dialog: BITCreateAssemDlg（文件——Add CAA V5 item——catia resource——dialog）,生成三个文件 BITCreateAssemDlg.h, BITCreateAssemDlg.cpp 和 BITCreateAssemDlg.CATDlg 在 BITCreateAssemDlg.CATDlg 中布局对话框。



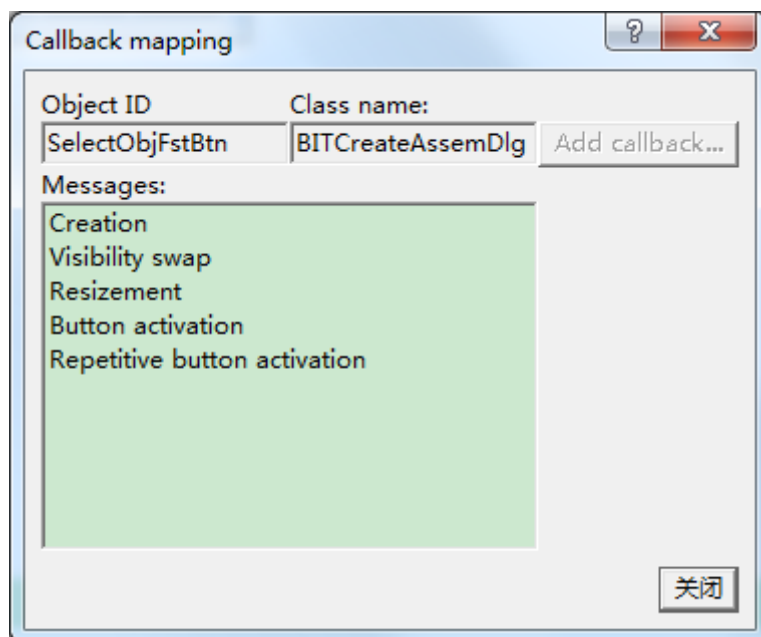
在窗体中添加三个按钮



为每个按钮更改属性中的 title 和 name，同上，Name 为程序中的引用时的代号，Title 为用户使用时界面上显示的名称。

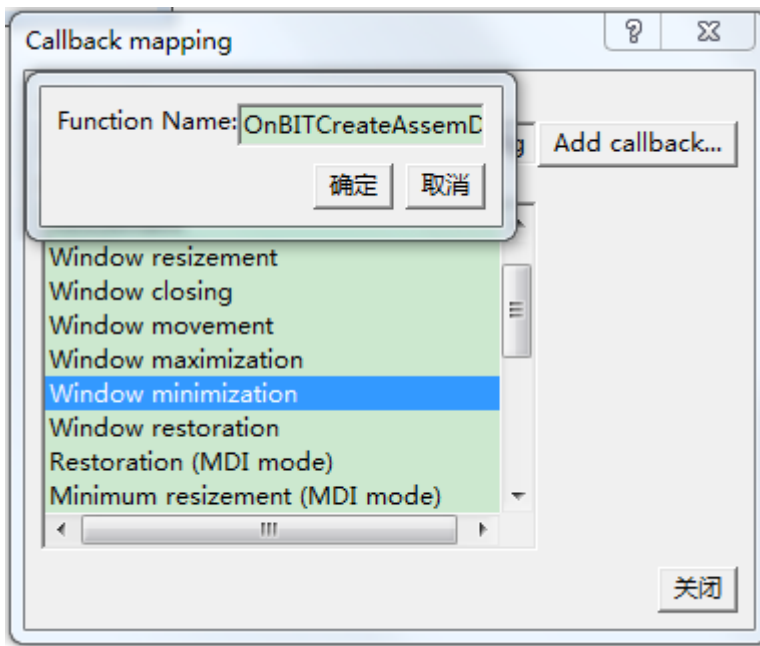


为每个按钮添加回调函数



均选择 Button activation.

同时为窗口上自动生成的最大化，最小化，关闭等回调函数。



添加完毕之后会在在 BITCreateAssemDlg.cpp 中自动生成代码，下面的表示按钮在对话框中的布局，可查询 SetGridConstraints 函数知道里面几个参数的含义。

```
CAA2 WIZARD WIDGET CONSTRUCTION SECTION
_SelectObjFstBtn = new CATDlgPushButton(this, "SelectObjFstBtn");
_SelectObjFstBtn -> SetGridConstraints(0, 0, 1, 1, CATGRID_4SIDES);
_SelectObjSndBtn = new CATDlgPushButton(this, "SelectObjSndBtn");
_SelectObjSndBtn -> SetGridConstraints(1, 0, 1, 1, CATGRID_4SIDES);
_CreateAssemBtn = new CATDlgPushButton(this, "CreateAssemBtn");
_CreateAssemBtn -> SetGridConstraints(2, 0, 1, 1, CATGRID_4SIDES);
//END CAA2 WIZARD WIDGET CONSTRUCTION SECTION
```

弹出的对话框基本上构建完成。

在 BITCreateAssemDlg.h 中添加头文件（后面用到，之后再添加也行）：

```
#include "CATISpecObject.h" //对象选择相关
#include "CATIPProduct.h" //操作 product 相关
```

在 BITCreateAssemDlg.cpp 中添加头文件（后面用到，之后再添加也行）：

```
#include "CATISpecObject.h"
```

现在到 BITCreateAssemCmd.cpp 中添加头文件: #include "BITCreateAssemDlg.h"

在 CATStatusChangeRCBITCreateAssemCmd::Activate 函数中添加如下代码：

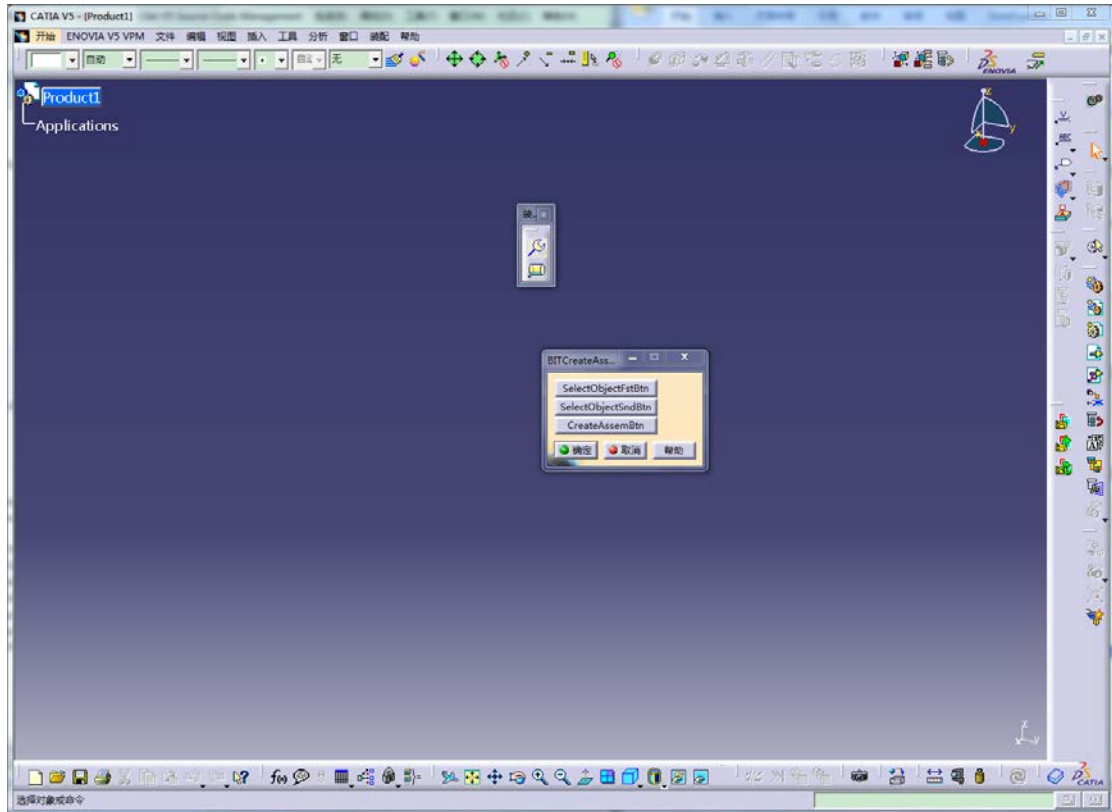
```
//命令响应菜单的单击操作，执行所需的命令
BITCreateAssemDlg* pDlg = new BITCreateAssemDlg();
pDlg->Build();
pDlg->SetVisibility(CATDlgShow);
RequestDelayedDestruction();
```

在 OK 按钮和 close 按钮回调函数中添加代码,点击这两个按钮时对话框消失:

```
this->SetVisibility(CATDlgHide);
```

```
this->RequestDelayedDestruction();
```

然后生成 mkmk 并启动 catia, 此时点击第一个装配按钮弹出对话框



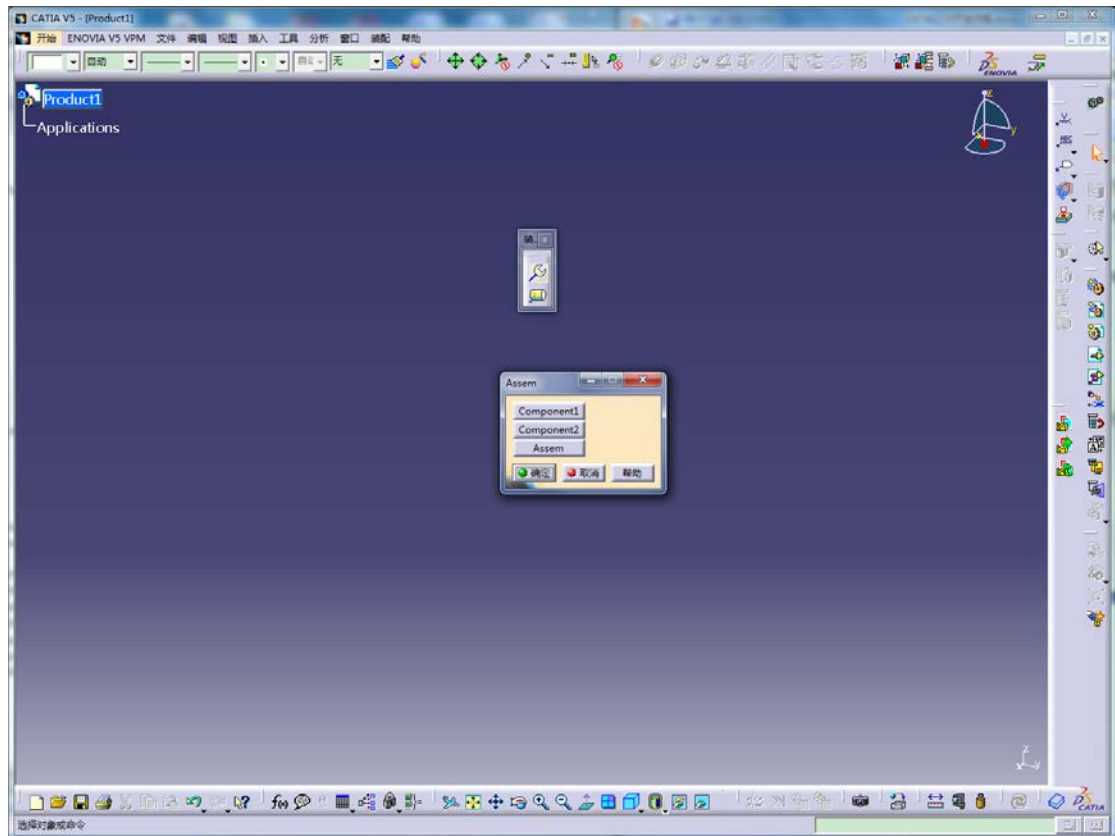
在 BITAssemFrmWork\CNext\resources\msgcatalog 中新建两个文档, 先新建文本文档然后修改后缀名, BITCreateAssemDlg.CATNls (存放显示的名称) BITCreateAssemDlg.CATRsc (存放图片等资源文件)。

在 BITCreateAssemDlg.CATNls 中添加:

```
// DO NOT EDIT :: THE CAA2 WIZARD WILL REPLACE ALL THE CODE HERE
Title = "Assem";
SelectObjFstBtn.Title = "Component1";
SelectObjSndBtn.Title = "Component2";
CreateAssemBtn.Title = "Assem";
// END WIZARD REPLACEMENT ZONE
```

BITCreateAssemDlg.CATRsc 中暂时不添加代码。

然后 Create/update runtime view 整合加载的模块的资源, 会看到按钮名称有所改变



## 2.2.4 为控价添加回调函数（Callback Wizard）

创建消息框

在 BITCreateAssemDlg.h 中添加

```
public:
CATDlgNotify * _OpenNotify;           //对话框CATDlgNotify的声明
```

在 BITCreateAssemDlg.CATDlg 中继续添加回调函数。

给按钮 Component1 添加回调函数。

在 BITCreateAssemDlg.cpp 中找到回调函数的框架：

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

在大括号中添加：

```
_OpenNotify = new CATDlgNotify(this, "", CATDlgNfyOK);
_OpenNotify->SetText("这是第一个消息框\n用全局变量_OpenNotify");
_OpenNotify->SetVisibility(CATDlgShow);
```

运行程序：

CAAV5 Workspace->Create/Update Runtime View->Ok

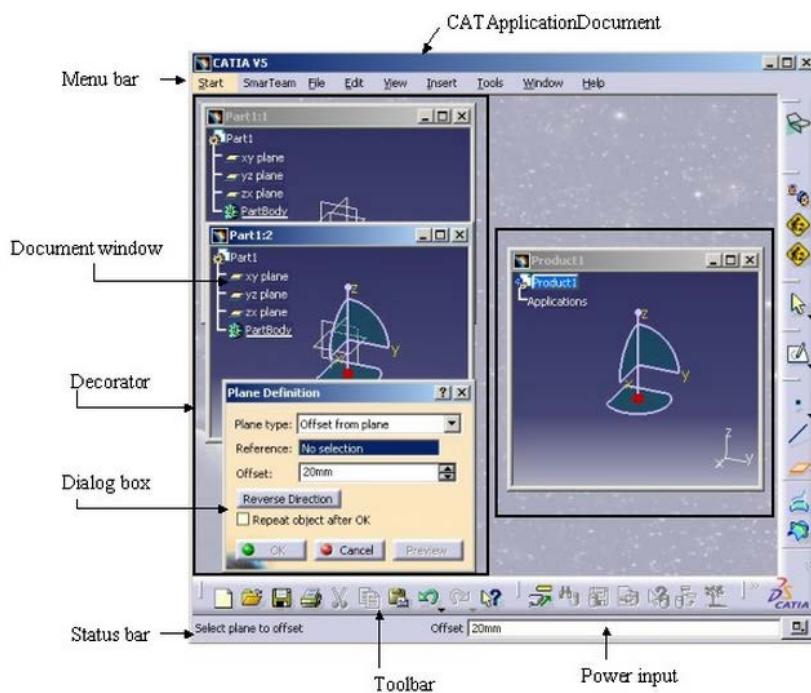
生成->mkmk->OK

窗口->Open runtime Window-输入”Cnext”->回车  
结果如下：



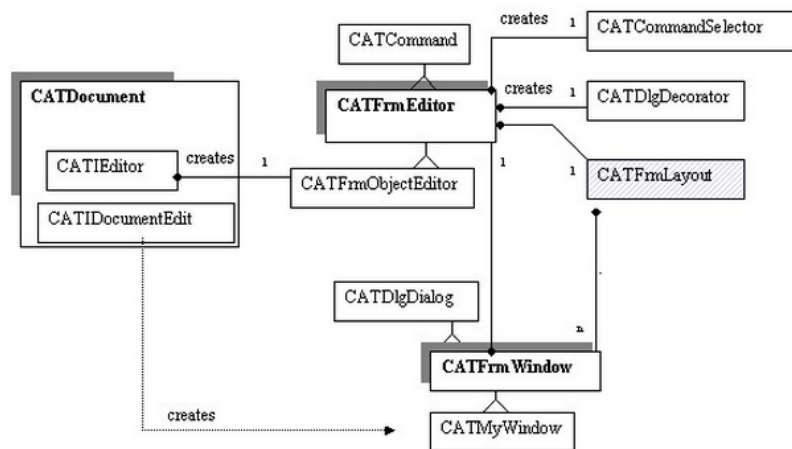
## 2.3 实例运用

### 2.3.1 对话框简介



- M = a CATDocument (document)
- V = a CATFrmWindow (window)
- C = a CATFrmEditor (editor)

Fig.3: The MVC Paradigm



Document 管理 editor;

window 通过两个接口创建：1.CATIEditor 用来创建 CATFrmEditor 实例和 CATFrmObjectEditor 实例。2.CATIDocumentEdit 用来创建窗口来显示 document。

CATFrmEditor 与布局有关的方法有 GetCurrentEditor ; GetDocument ; GetWindowCount ; GetCommandSelector。

### 2.3.2 读取 txt 文件

```

//-----
// Callback on PushBActivate of _PushButton001
//-----

virtual void OnPushButton026PushBActivateNotification (CATCommand *, CATNotification*, CATCommandClientData
data)
{
    _Editor021->ClearLine();
    _Editor022->ClearLine();
    _Editor023->ClearLine();
    _Editor024->ClearLine();

    //开始读取文档内容

    ioText Read;
    HRESULT rc=E_FAIL;

    rc=Read.Open(Path);

    CString Content[40]; //存放文本的内容
    if (!SUCCEEDED(rc))

```



---

```
{
CATDlgNotify* pNotif = new
CATDlgNotify(this, "AAA", CATDlgNfyInformation|CATDlgWndModal);
pNotif->DisplayBlocked ( "设计文档打开失败", "提示");
}
```

```
else
```

```
{
int i;
Read.ReadText();

for (i=1;i<=Read.AllText.Size();i++)
{
CATUnicodeString temp;
CString TM1, TM2;
TM1=Read.AllText[i].ConvertToChar();

int Len=TM1.Find(' ');
int NumOfNumber=Len+3;
int LenghtOfCString=TM1.GetLength();
TM2 = TM1.Right(LenghtOfCString-NumOfNumber);
Content[i-1]=TM2;
}

Read.Close();
}
```

```
//转成CATUnicodeCString
```

```
CATUnicodeString temp0;
wchar_t * wchar0;
wchar0= Content[0].GetBuffer(Content[0].GetLength());
temp0.BuildFromWChar(wchar0);
CATUnicodeString temp1;
wchar_t * wchar1;
wchar1= Content[1].GetBuffer(Content[1].GetLength());
temp1.BuildFromWChar(wchar1);
CATUnicodeString temp2;
wchar_t * wchar2;
wchar2= Content[2].GetBuffer(Content[2].GetLength());
temp2.BuildFromWChar(wchar2);
CATUnicodeString temp3;
wchar_t * wchar3;
wchar3= Content[3].GetBuffer(Content[3].GetLength());
```

---

```
temp3.BuildFromWChar(wchar3);
```

```
//赋值
```

```
_Editor021->SetText(temp0);  
_Editor022->SetText(temp1);  
_Editor023->SetText(temp2);  
_Editor024->SetText(temp3);  
  
}
```

### 2.3.1 创建几何特征（Point）和实体特征（Solid）

```
//-----  
// Callback on PushBActivate of _PushButton001  
//-----  
void FirstDlg::OnPushButton001PushBActivateNotification(CATCommand* cmd, CATNotification* evt,  
CATCommandClientData data)  
{  
    // Add your code here  
}
```

在回调函数的大括号内中写入以下代码：

```
//////////获得Editor、Document、Container、设置GSMFactory  
CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();  
CATDocument *pDoc = pEditor->GetDocument();  
CATIContainerOfDocument_var spCon0Docs = pDoc;  
CATIContainer* _pContainer; //Container  
CATIGSMFactory_var spGSMFactory; //GSM工厂  
_pContainer = NULL;  
HRESULT hr = spCon0Docs->GetSpecContainer(_pContainer);  
spGSMFactory = NULL_var;  
spGSMFactory = _pContainer;  
//设置点的坐标  
CATMathPoint _Point;  
_Point.SetCoord(10, 10, 10 );  
//////////用以上得到的Point3D画点  
CATIGSMPoint_var spPoint1 = spGSMFactory->CreatePoint(_Point); //创建一个点  
CATISpecObject_var spSpecPoint1;  
spSpecPoint1 = spPoint1; //Casts the point as a CATISpecObject  
CATIGSMProceduralView_var spSndPntObj = spSpecPoint1;  
spSndPntObj->InsertInProceduralView();
```

找到回调函数位置并在按钮的回调函数中输入以下代码：

```
void FirstDlg::OnPushButton002PushBActivateNotification(CATCommand* cmd, CATNotification* evt,  
CATCommandClientData data)  
{    //创建草图  
    //////////获得Editor、Document、Container、设置GSMFactory
```

---

```

CATFrmEditor* pEditor = CATFrmEditor::GetCurrentEditor();
CATDocument *pDoc = pEditor->GetDocument();
CATIContainerOfDocument_var spCon0Docs = pDoc;
CATIContainer*          pContainer;          //Container
CATIPrtContainer*       pPrtContainer;       //PartContainer
CATIGSMFactory_var      spGSMFactory;        //GSM工厂
CATIPrtPart_var         spPrt              ;//
HRESULT hr = spCon0Docs->GetSpecContainer(pContainer); //获取Container
hr = pContainer->QueryInterface(IID_CATIPrtContainer, (void **)&pPrtContainer); //获取PrtContainer
spPrt = pPrtContainer->GetPart();
CATListValCATISpecObject_var spListRefPlanes = spPrt->GetReferencePlanes(); //获取个参考平面
CATISpecObject_var spSketchPlane = spListRefPlanes[3]; //第三个平面
CATISketchFactory_var spSketchFactOnPrtCont(pPrtContainer); //草图工厂
CATISketch_var spSketch = spSketchFactOnPrtCont->CreateSketch( spSketchPlane ); //创建草图
CATI2DWFFactory_var spWF2DFactOnSketch(spSketch);
spSketch->OpenEdition(); //进入草图开始绘图
double centerPoint[2]={0.0,0.0};
CATISpecObject_var spCircle = spWF2DFactOnSketch->CreateCircle(centerPoint, 10); //绘制一个圆
spSketch->CloseEdition(); //退出草图环境
CATIPrtFactory_var spPrtFactOnPrtCont(pPrtContainer); //零件工厂
CATISpecObject_var contPad = spPrtFactOnPrtCont->CreatePad(spSketch); //创建拉伸
CATIPad_var ospad = contPad;
ospad->ModifyEndType(catOffsetLimit); //设置拉伸高度为20mm
ospad->ModifyEndOffset(20.00);
contPad->Update(); //实体更新
// Add your code here
}

```

以上创建点和创建圆柱的过程中使用了很多 CAA 的类，这些类在使用之前必须声明。所以必须在使用之前加上头文件：

```

#include "CATFrmEditor.h" // CATIA的FrameEditor
#include "CATDocument.h" //CATIA 的文件
#include "CATIContainerOfDocument.h" //CATIA 的文件的Container
#include "CATIGSMFactory.h" // geometrical elements of Generative Shape Design workshop
#include "CATIGSMPoint.h" // point feature
#include "CATIGSMProceduralView.h"
#include "CATIContainer.h"
#include "CATISpecObject.h"
#include "CATIPrtContainer.h"
#include "CATISketchFactory.h"
#include "CATIPrtFactory.h"
#include "CATIPrtPart.h" //用来获取参考平面
#include "CATISketch.h" //用来创建草图
#include "CATI2DWFFactory.h"
#include "CATIPrism.h" //拉伸棱柱

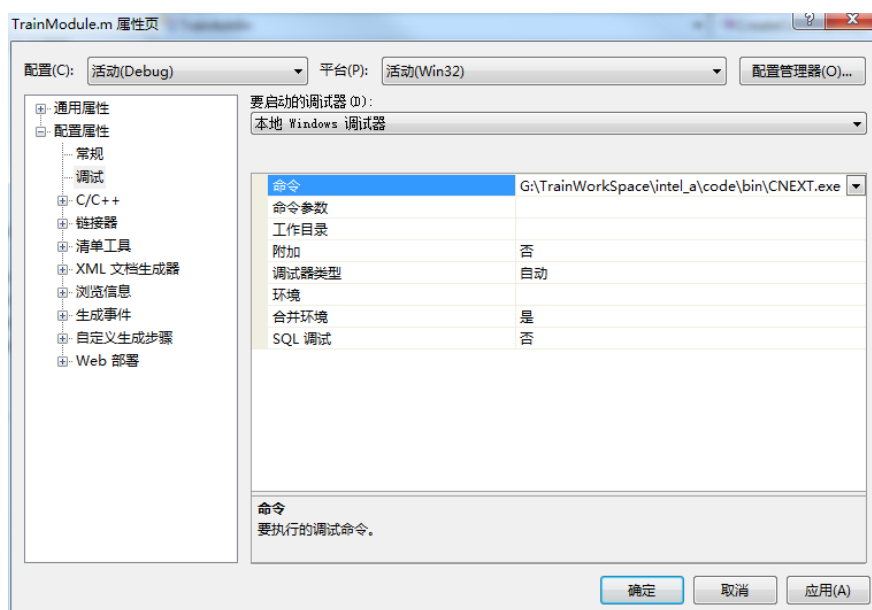
```

```
#include "CATIPad.h" // 凸台
#include "CATLimitDefs.h" // 长度定义
```

### 3 编译和调试

(1) 设置工程的调试命令文件。右键点击添加的模块，在属性页中选择配置属性，点击调试，在右边的命令中找到项目生成的 CNEXT.exe。位置在项目空间路径 \intel\_a\code\bin\CNEXT.exe。点击应用，确定关闭对话框。

依次设置其他的 module 和 Framework。



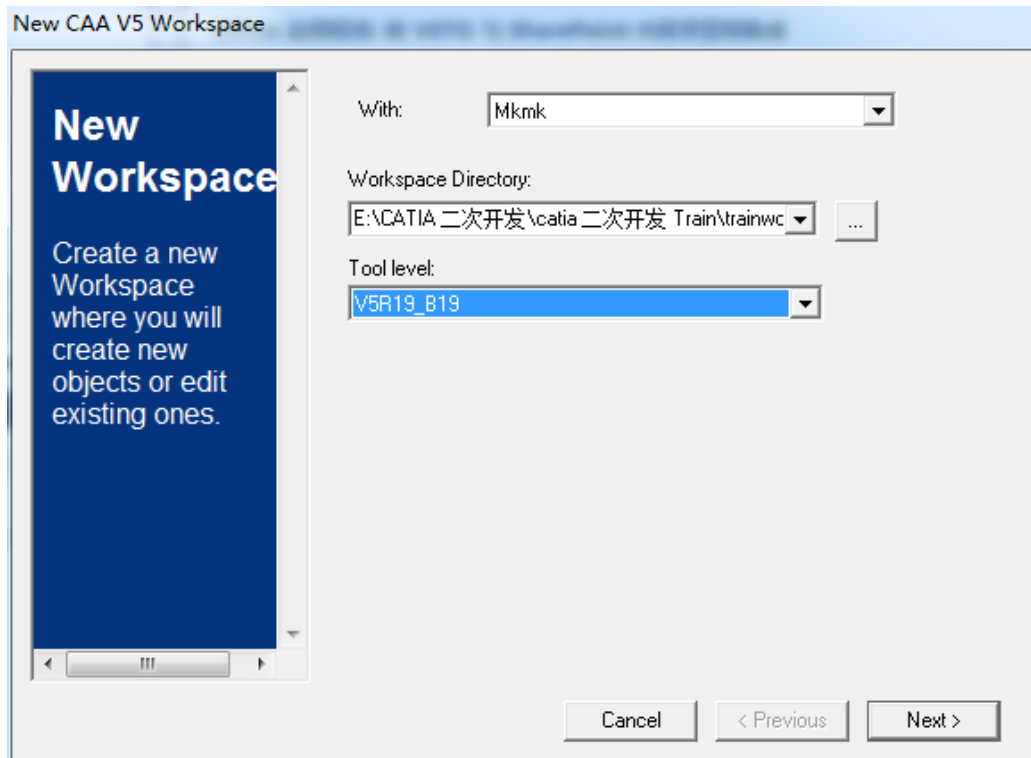
(2) 在程序中添加断点。注意断点不要加在注释行和空行。

(3) 点击调试菜单中“启用调试”。

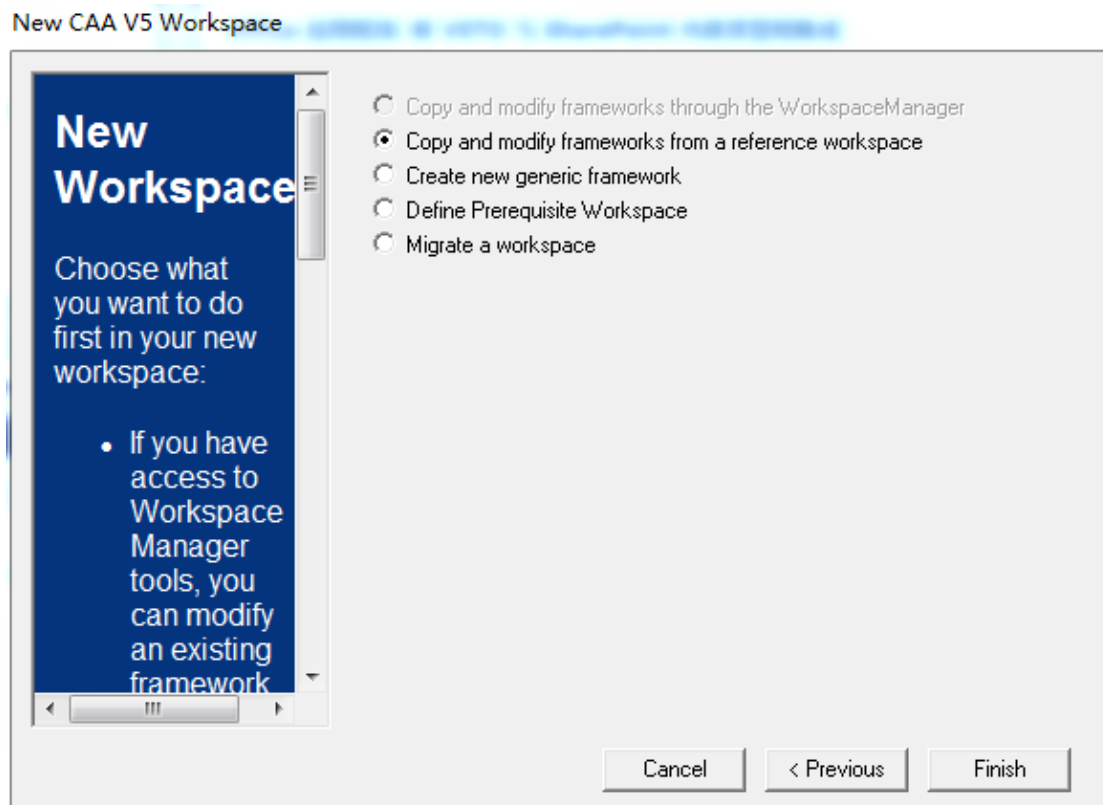
### 4 复制工作空间

当个工作空间打不开的时候，或者要更换平台的时候可以复制一个工作空间。首先新建一个文件夹作为工作空间；

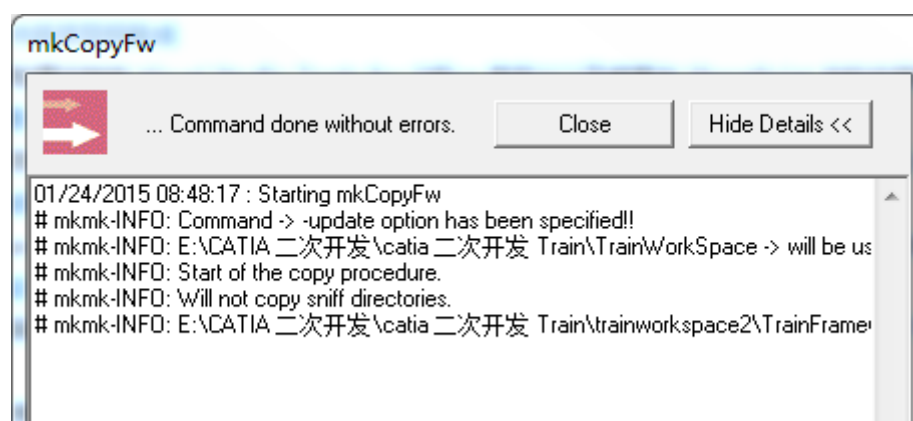
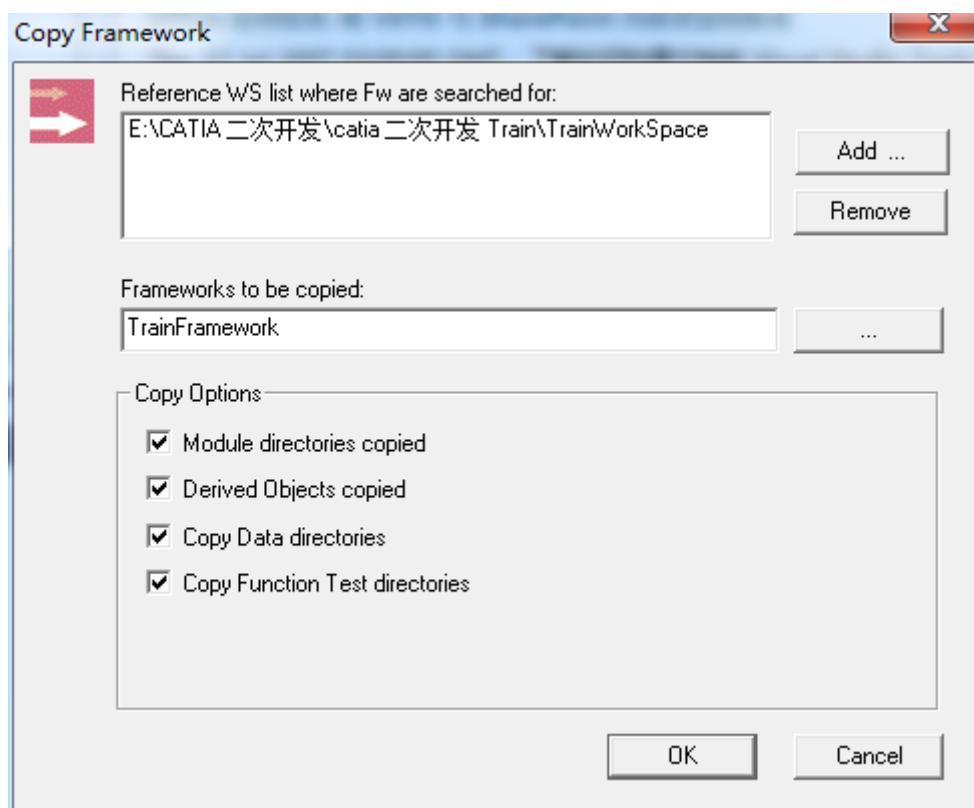
在 VS 中点击文件-> new caa V5 Workspace;在 Workspace directory 中选择新建的文件夹，tool level 选择正确。进入下一个页面。



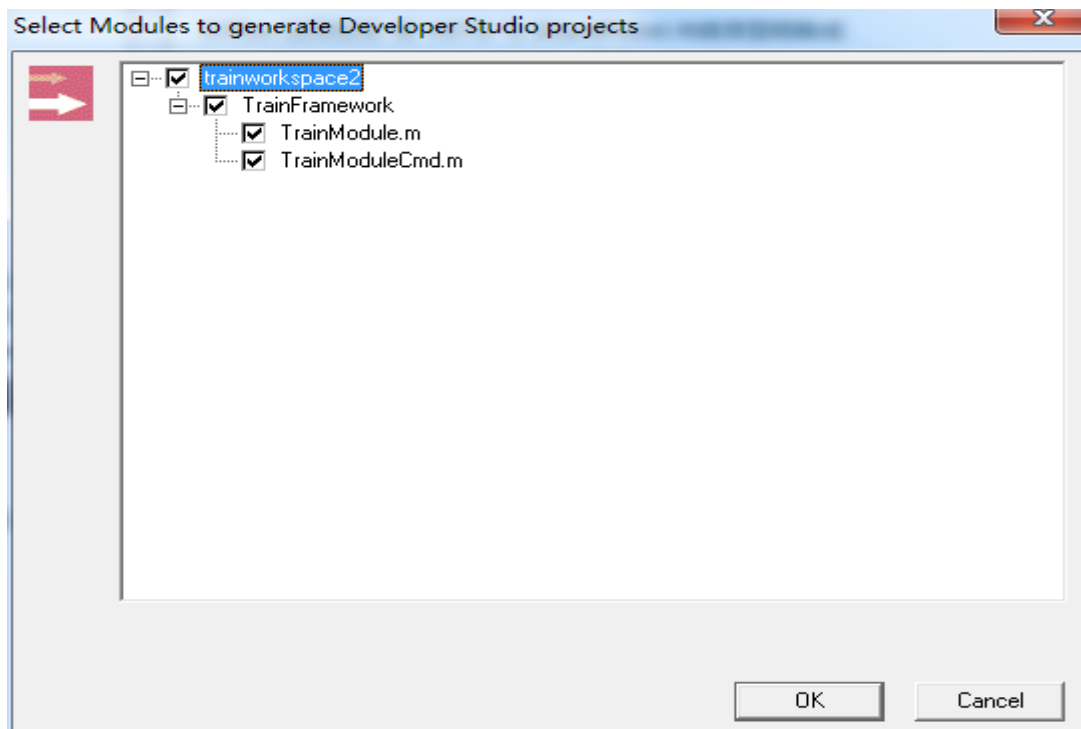
在这个页面选择 copy and modify XXXXXX； 点击 finish;



在下一个页面点击 add 路径选择到你要复制的那个工作空间,再点击...选择 framework 在 copy options 中勾选四个选项。点击 OK。



点击 Close 后出现这个页面，勾选所有内容。点击 OK。



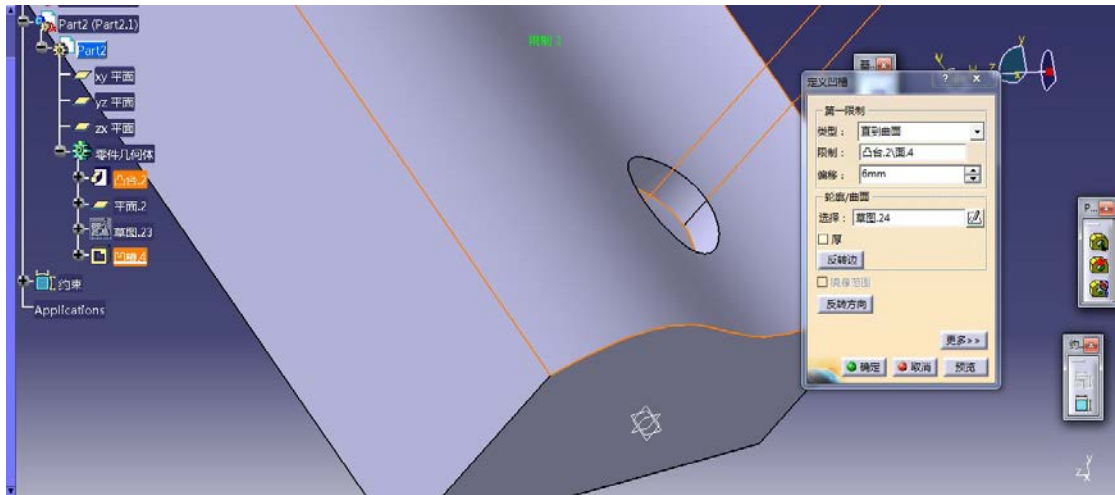
## 5 总结

Workbench , workshop , workspace , Framework , module, menu bar, tool bar, dialog box, command, addin

### 注意 1:

在百科全书和帮助文档中查找相应的函数的时候有可能找不到。帮助文档中并没有列出全部的方法。这时可以在 CAA 安装目录寻找头文件，从头文件中查找你要的方法。头文件中的方法肯定是全部方法。

比如查找 `CATIPrism::ModifyEndoffsetFromSurface()`：它的功能为设置拉伸或者凹槽直到曲面方式中的偏移。这个方法在百科全书中没有列出。但是在头文件 `CATIPrism.h` 中可以找到的该方法。



CAA 的安装中的文件夹中搜索 CATIPrism.h 即可。打开 CATIPrism.h 文件，可以看到全部方法：

```

CATIPrism.h  CATIPocket.h  TSlot.cpp  CrossSlotDlg.h  CrossSlotDlg.cpp  CAAMmrRetrieveCo...ectorsFromPad.cpp
(未加载图)
121  * @nodoc
122  * Do not use - Not yet implemented.
123  */
124  virtual void ModifyEndOffsetFromSurface(double iOffset) = 0;
125
126  /**
127  * Sets the second limit type.
128  * @param iType
129  * The type.
130  * <br><b>Legal values</b>: They are provided with the <tt>CatLimitMode</tt> enum.
131  * @see CATIPrtSimpleLimit
132  */
133  virtual void ModifyStartType(int iType) = 0;
134
135  /**
136  * Sets the second limit offset value in case the type is <tt>catOffsetLimit</tt>.
137  * @param iOffset
138  * The offset value.
139  */
140  virtual void ModifyStartOffset(double iOffset) = 0;
141  ...

```

## 注意 2：

初学者会经常遇到 link 2001 等形式的链接（Link）错误，这种错误绝大部分是由于 Imakefile.mk 或者 IdentifyCard 出错造成的，此时修改这两个文件，或者直接从 train1 的源程序中找到该文件，并将其复制过来。

## 注意 3：

创建圆柱的程序必须要在 Part 环境运行，否则会出现“单击确定终止”字样的崩溃提示。

其实“单击确定终止”的报错绝大部分是由于指针操作有误造成的。这种时候就可以采用调试流程进行程序调试，找到程序崩溃的地方，然后仔细检查程序，解决问题。

## 注意 4：

在社区 Help CAA V5 中不能打开帮助文档时可以在百科全书中找到帮助文档，依次点击 C++ API，->class Hierarchy。在页面中搜索可以产生相同的检索效果。





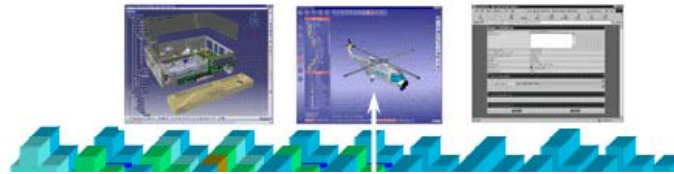
[C++/Java Home](#)  
[Automation Home](#)  
[Web Services Home](#)

CAA V5 Encyclopedia

[Site Map](#) | [Getting Started](#) | [What's New?](#) | [FAQ](#)

[Search](#)  
[C++ API](#) | [Java API](#)  
[Automation API](#)

CATIA  
DELMIA  
ENOVIA



[C++/Java Home](#)  
[Automation Home](#)  
[Web Services Home](#)

CAA V5 Encyclopedia

[Site Map](#) | [Getting Started](#) | [What's New?](#) | [FAQ](#)

[Search](#)  
[C++ API](#) | [Java API](#)  
[Automation API](#)

All frameworks [Class Hierarchy](#) [Indexes](#)

## All Frameworks

- Framework [Administration](#)
- Framework [AdvancedMachiningInterfaces](#)
- Framework [AdvancedMathematics](#)
- Framework [AdvancedTopologicalOpe](#)
- Framework [AnalysisMeshingModel](#)
- Framework [AnalysisMeshingToolsItf](#)
- Framework [ApplicationFrame](#)