ICS 25.040
J 07

中华人民共和国国家标准

GB/T 18726—2002

现代设计工程集成技术的软件接口规范

Software interface specification of integrated technology of modern design engineering

2002-05-20 发布　　2002-12-01 实施

中华人民共和国
国家质量监督检验检疫总局　发布

目　　次

前　言

本标准采用目前国际上流行的面向对象的中间件技术，定义了符合我国实践特点的现代设计工程集成技术的软件接口规范。本标准是在我国制造业实施 CIMS 工程改造多年来的实践基础上研究制定的，它将为我国企业和现代设计工程相关领域的软件提供商提供一个符合现代软件接口发展潮流的执行规范，使得越来越多的与现代设计工程相关的软件可以实现实时、动态的数据和功能交换，使得企业的设计信息系统具有更高的集成性，提高产品设计与管理的效率。

本标准在制定时参照了国际上广泛应用的面向对象的中间件设计技术 CORBA 和 COM，这是我国制造业目前应用最为广泛的面向对象的中间件技术。标准所涉及的范围主要是我国近年来得到广泛应用的与设计工程相关的信息技术 CAD、CAPP 及 PDM 等。

本标准的附录均为提示的附录。

本标准由中国机械工业联合会提出。

本标准由全国工业自动化系统标准化技术委员会归口。

本标准起草单位：北京机械工业自动化所。

本标准参加起草单位：北京航空航天大学、北京北航海尔软件有限公司、中国科学院沈阳自动化所、航天部二院质量标准处、北京艾克斯特工业自动化技术有限公司、北京京渝天河计算机软件有限公司、保定天威集团大型变压器公司、广州红地技术有限公司。

本标准主要起草人：王涛、刘爱军、马健、陈小慧、曾宇波、孙东光、李宁。

引　　言

0.1　面向对象的软件中间件设计方法

中间件是一个将数据与功能封装在一起以完成特定任务的计算机程序，它本身往往不能单独运行，要在其宿主程序中与其他程序一起协调地工作。中间件把应用程序与系统所依附软件的较低层细节和复杂性隔离开来，使应用程序开发者只处理某种类型的单个 API——其他细节则由中间件处理。这种将接口与实现分离的好处是可以采用灵活的、积木式的开发方法。例如可以通过 OLEDB 对数十种不同的数据库进行访问，OLEDB 就充当了数据库应用程序和数据库系统之间的中间件。

面向对象的中间件为不同的开发语言建立了一个标准的数据和功能调用的方法。这使得不同软件开发商的产品可以通过各自提供的中间件交换数据和功能。

0.2　面向对象的软件接口设计方法

不难想象，如果将中间件技术作为技术信息中各系统之间的信息通讯手段——各系统均提供其他系统所需的处理本系统相应数据的通讯中间件，就可以使技术信息的集成达到一个动态、实时的新水平，它不仅可以使技术信息各相关系统可以共享必要的信息，甚至可以使它们共享所需的功能和应用界面。这就是本项目的目的——定义并促使与技术信息相关的不同软件系统提供并开放信息交流的中间件。这里应强调的是，要定义的中间件是可用于软件编程的标准方法，而主要不是定义要交流的信息格式。

0.3　现代设计工程软件的集成方法

目前在制造业技术信息领域，流行的面向对象的中间件标准有两个，微软的 COM、OMG（对象管理集团）的 CORBA，基于它们开发出来的中间件都具有可重用性和扩展性。显然将这些标准与其他基于文件的信息交换标准相结合，制定出设计信息系统中各子系统或模块应遵循的中间件接口标准，将使标准化工作从基于文件的信息交换推广到软件运行的实时水平，从而使标准化工作跃上一个新台阶，这是符合当前发展潮流的。

本规范所提供的集成接口方法示意图如图 1 所示。由于各单元应用系统除提供了操作界面供用户使用本功能外，还提供了和其他系统交换数据的动态接口（面向对象的中间件），使得接口的提供成为相应软件系统的标准配置，同时该系统也成为其他系统提供数据和功能的服务器。由于面向对象的中间件的特性，这些接口具有与语言无关性，它们的可继承性使得接口可以随着软件的升级而升级，并且具有向下的兼容性，客户系统不需要理解要连接的服务器系统的内部数据格式，只要通过服务器提供的中间件接口就可实现动态的处理服务器提供的数据和使用服务器的功能。这种方法在很大程度上可以实现软件系统间的无缝集成。

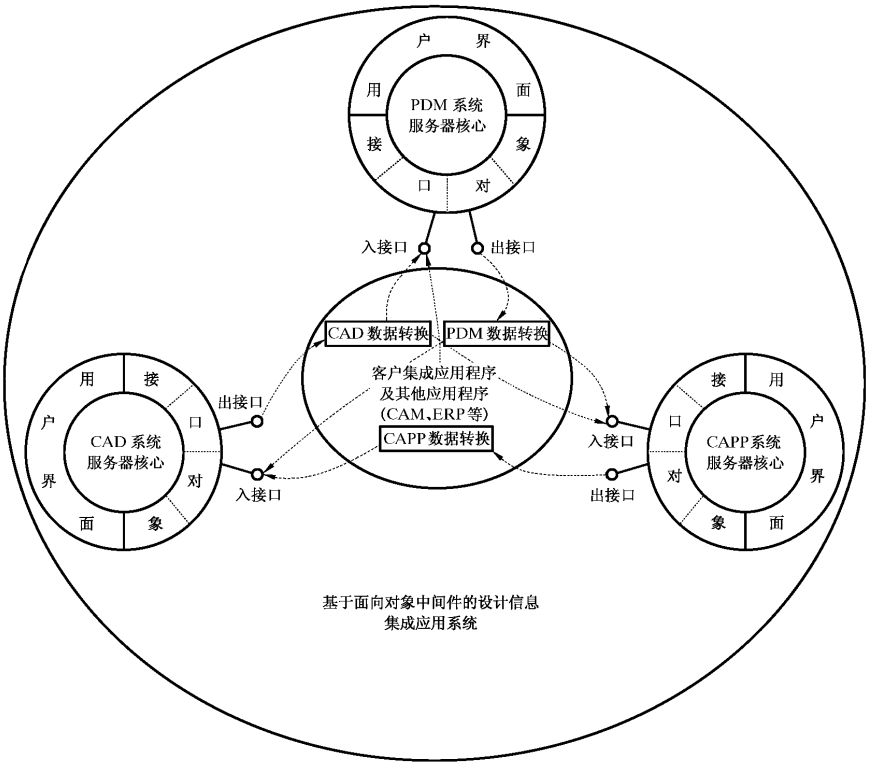

图 1　基于面向对象中间件的设计信息集成接口示意图

0.4　本标准的作用

在我国 CIMS 应用正在广泛展开之时，本标准的制定将有深远的意义。主要体现在以下几个方面：

a）大大提高我国制造业技术信息集成的水平和步伐。得到实惠最大的将是应用企业，他们在信息集成方面投入的成本和时间将大为缩短，水平将大为提高。企业在技术信息各方面可以更加放心地对所需的软件进行优化组合，不会由此而过分担心集成问题。

b）软件供应商，特别是我国的软件供应商将从规范中获得许多利益。首先，由于技术信息软件领域已进入一个快速细分和重组的过程，过去由一家包揽一个企业技术信息各方面软件及实施的方式已经过时，软件提供商会在自己不擅长的领域更容易地与他人合作。但是如果针对每一个企业，软件商们都要为其定制特殊的接口，这样做不仅耗费大量的时间和精力，效果也难以达到最佳。本规范的实施将为软件提供商们带来更大的市场空间，也将为国内的技术信息咨询和集成领域的规范化提供契机。

c）本标准的实施，将促使国内软件供应商大幅度提高其软件的设计水平。采用中间件被称为软件开发技术的一场革命，甚至有人认为软件开发技术到现在才进入成熟阶段。而我国软件企业在这方面有深入实践的还不多见。毫无疑问，标准的实施将使我国技术信息领域的软件供应商步入现代化的软件开发方式。

中华人民共和国国家标准

现代设计工程集成技术的软件接口规范

GB/T 18726—2002

Software interface specification of integrated technology of modern design engineering

1 范围

本标准定义并促使与技术信息系统(TIS)相关的不同软件系统提供并开放信息交流的中间件。这些软件系统主要可分为以下几类:PDM、CAD、CAPP 等,开放的中间接口可用于支持与现代设计工程相关的许多其他软件系统,如 CAE、CAM 等,也可支持如 ERP 等管理信息系统。

本标准适用于指导企业在实施技术信息集成过程中,定义不同软件的动态接口或在选购相关软件时作为判断其可集成程度的参考。

2 引用标准

下列标准所包含的条文,通过在本标准中引用而构成为本标准的条文。本标准出版时,所示版本均为有效。所有标准都会被修订,使用本标准的各方应探讨使用下列标准最新版本的可能性。

GB/T 17304—1998 CAD 通用技术规范

3 定义

本标准采用下列定义。

3.1 技术信息系统 Technique Information System(TIS)

技术信息主要是描述企业产品设计阶段所产生出来的有关产品定义、设计、设计过程等的相关信息。例如产品的设计数据(图纸、计算书、设计说明等)、工程分析数据、工艺数据、设计流程数据等。

管理技术信息的软件系统就称为技术信息系统。

3.2 产品数据管理 Product Data Management(PDM)

对整个产品生命周期内的产品设计、制造数据及产品管理数据进行的管理。

3.3 计算机辅助设计 Computer-Aided Design(CAD)

包括绘图与说明的设计活动,其中信息处理系统用于完成对一个零件或产品功能的设计与改进。

3.4 计算机辅助工艺设计 Computer-Aided Process Planning(CAPP)

在产品制造过程中,利用计算机辅助编制工艺计划,如工艺路线卡和检验工序卡等。

3.5 计算机辅助制造 Computer-Aided Manufacturing(CAM)

一个生产过程,其中信息处理系统用来指导与控制制造。

3.6 中间件 middleware

一个将数据与功能封装在一起以完成特定任务的计算机程序,它本身往往不能单独运行,要在其宿主程序中与其他程序一起协调地工作。中间件把应用程序与系统所依附软件的较低层细节和复杂性隔离开来,使应用程序开发者只处理某种类型的单个应用接口,其他细节则由中间件处理。这种将接口与

中华人民共和国国家质量监督检验检疫总局 2002-05-20 批准　　2002-12-01 实施

实现分离的好处是可以采用灵活的、积木式的开发方法。

3.7 面向对象的中间件 object-oriented middleware

具有面向对象软件设计特性的中间件，这些特性包括封装、多态性、动态联编、继承性等。面向对象的中间件还具有和语言的无关特性——即无论一个中间件是采用什么语言实现的，支持这种中间件的任何其他语言都可以在运行代码的级别上使用它，而不是使用其源码。

3.8 对象管理集团 Object Management Group(OMG)

由众多与信息技术相关的公司所组成的开放的研讨组织，其工作是制定对象计算的开放标准。CORBA 就是它制定的一个分布式环境下的重要的面向对象的中间件标准。

3.9 公共对象请求代理体系 Common Object Require Broker Architecture(CORBA)

由 OMG 制定的面向对象的中间件标准。

3.10 组件对象模型 Component Object Model(COM)

由 Microsoft 提出的面向对象的中间件标准，它定义了中间件程序之间进行交互的标准和所需的运行环境。

3.11 接口定义语言 Interface Define Language(IDL)

用于描述中间件接口特性和功能的计算机语言。

3.12 微软接口定义语言 Microsoft Interface Define Language(MIDL)

Microsoft 在 OSF 的 DCE 规范 IDL 的基础上扩充的用于描述 COM 接口的 IDL。

3.13 ActiveX

实现微软 COM 标准的一组技术，它使得采用该技术的软件不仅在本地计算机而且可以在国际互联网上动态实现信息和功能的互操作。它定义的技术主要包括：ActiveX 控制(ActiveX Control)，ActiveX 文档(ActiveX Document)、ActiveX 组件(Active Component)等。

3.14 自动化对象 automation object

COM 对象的一个特例，它简化了 COM 的一些底层细节，它通过特定的 IDispatch 接口提供了一些弱类型语言实现 COM 对象的方法。

3.15 工艺数据 process data

在产品工艺设计过程中产生的数据，虽然在不同的 CAPP 系统或相关的系统中会有不同的定义，其浏览方式也会不同，但是工艺数据的分类是大致明确的。例如在机械制造业中，工艺数据可以分为：工艺路线、工艺卡片(不同的工艺类型会有不同的工艺卡片)、工艺统计信息(如工艺过程所需的工装、毛坯材料)等。

3.16 加工单元 process unit

一个抽象的概念，某个部件、产品的工艺路线、某个零部件的不同专业的工艺过程(如机加工工艺，热处理工艺，装配工艺)、以及加工过程中的不同工序都可以作为加工单元处理。

3.17 加工单元集 process units

指服务于某一特定工艺目的的一组相关操作的加工单元的集合。例如一个零件有许多不同专业的加工工艺。另外加工单元之间也存在从属关系，例如：不同专业的加工过程是从属于工艺路线的，工步是从属于工序的。

3.18 过程记录集 process record set

加工单元的主要内容，用来全面描述某个工艺路线、或加工过程的所有步骤。过程记录集中反映的是一个二维数据表的数据，该表中的每一条记录对应加工过程中的某个步骤。过程记录集是在工艺设计过程中产生的。

3.19 工艺卡片 process card

工艺设计最终结果的表现形式，有的工艺卡片是以 CAD 图形格式存储的，主要用于浏览和打印。

3.20 对象定义语言 Object Define Language(ODL)

Microsoft MFC 所采用的用于定义 COM 接口对象的定义语言，它是 IDL 的扩展。

4 软件集成接口描述

4.1 接口定义方法和原则

4.1.1 接口定义的形式

本标准定义了现代设计工程中相关软件系统应提供的接口内容及定义应遵循的原则，但不定义该接口的具体实现。作为参考，规范给出了一些接口的实现实例。

本标准要求所有被定义开放的接口均使用面向对象的中间件技术。

4.1.2 接口定义应描述的内容

每一接口的描述应包括如下内容：

a) 接口功能描述，可实现的接口级别；

b) 接口的运行环境，包括硬件和软件环境；

c) 接口定义所采用的语言及接口实现的语言工具；

d) 可用于使用接口的语言工具；

e) 接口定义；

f) 用于测试接口的实例。

本标准对接口定义了级别，该级别可用于判断软件可提供接口的水平。一般来说，级别越高的接口，要求系统的开放程度也越大，但实现的难度不一定大。

4.1.3 接口定义语言

制造业流行的面向对象的接口定义语言大部分是在 OSF(开放式软件基金会)的 DCE(分布式计算环境)标准中制定的 IDL 的基础上发展而来的。例如 OMG 的 IDL 以及 Microsoft 的 MIDL。因此在本标准的实现中，推荐使用基于 DCE 的 IDL。软件开发者所使用的接口开发工具，可以采用相应的扩展 IDL。

4.1.4 接口级别定义

接口定义的级别主要依据以下几种原则进行划分：

第一级：可以提供浏览服务器数据的能力。例如：CAD 系统浏览器接口。

第二级：可以提供提取服务器数据的能力。例如：CAD 标准格式文件的转换器接口、工程图标题栏、明细表数据的提取工具等。

第三级：可以提供存取服务器数据的能力。例如：CAD 系统提供的在其系统外部修改工程图标题栏、明细表等管理数据的能力。修改数据一般是服务器系统的基本权利，但是如果可以开放其中的部分接口，则可以促使不同系统之间达成更加完美的集成，所以将本级接口定为推荐采用接口。

第一级及第二级要求服务器系统提供其数据的出接口，第三级要求服务器系统提供其数据的入接口。

一个服务器系统，并不一定在满足了所有的低级接口后才能满足高级接口。它可以在某些接口上满足低级接口，而在其他接口上满足高级接口。

4.1.5 接口信息分类

按照接口信息在设计过程中出现的不同阶段，规范将软件接口信息分为三类：产品数据管理软件开放接口、计算机辅助设计软件开放接口、计算机辅助工艺设计软件开放接口。这种分类并不具体对应软件供应商提供的相应系统：PDM 软件系统、CAD 软件系统、CAPP 软件系统。软件供应商可以在其提供的系统中将开放的接口信息按照自己的数据结构定义，放在不同的软件系统中。例如供应商可以将工艺统计数据放在 PDM 系统中，将工装设备统计数据放在 CAM 系统中等。但是为了符合本标准，无论这些信息在哪个软件系统中定义或管理，它们都应在其被定义的系统中以面向对象的中间件形式开放接口。

4.2 产品数据管理软件开放接口定义

本条定义的接口主要涉及到企业的产品设计数据的共享和操纵。这些数据目前主要由 PDM 软件

或相关的软件进行处理。这里定义的接口涉及到产品数据的两个方面：产品的定义数据及设计过程中的管理数据。

4.2.1 产品数据浏览器。接口级别：一级

本接口使得客户程序可以浏览 PDM 系统中的产品定义数据。该数据一般是以产品结构树形式派生的。该浏览器一般提供如下功能：

a）产品的装配或配置结构，该结构表达了产品的各组成部件和零件的装配或隶属关系；

b）产品属性，它定义了产品的实用、制造及管理、性能等方面的数据；

c）产品文档（产品文档的构成可能是很广泛的，在不同的 PDM 系统中可能会不同），例如有的 PDM 系统可以浏览和产品制造有关的信息；

d）产品的其他相关内容，例如产品设计任务书、更改通知单等。由于不同的产品数据管理软件对产品数据的表达范围和方式可能不同，因此各软件所开放的接口内容也可以不同。

接口参考实例：见附录 A。

4.2.2 设计过程数据浏览器。接口级别：一级

本接口使得客户程序可以浏览 PDM 系统中的过程数据，一般包含如下功能：

a）浏览项目结构，项目结构是组织设计任务的一种层次化的方式，在不同 PDM 系统中有不同的定义方法，采用本接口可以显示出 PDM 系统定义的项目结构；

b）浏览任务属性，任务属性是描述任务基本配置的一组信息，包括负责人、时间、要求、资源、流程、状态等，在各 PDM 系统中有不同的数据项；

c）浏览任务文档，设计的过程是不断产生文档的过程，对任务文档的管理将为产品数据的管理提供可控的、确认后的规范化数据，由于设计是一项反复修正、迭代的工作，故任务可能是有版本的。

d）浏览任务评审数据，过程数据是经过工作流程驱动后形成的规范化数据，在工作流程当中，将产生各种任务的评审数据，如：评审意见、批注文件等，是对任务的一种补充性的经验数据，在不同 PDM 系统中有不同的处理方法。

接口参考实例：见附录 B。

4.2.3 产品数据提取接口。接口级别：二级

本接口使得客户程序可以按照服务器系统给定的格式提取产品的数据。这些数据可以是产品结构、产品属性、技术要求、产品文档、产品更改等（其中产品结构提取接口在 4.2.4 详细说明），数据的定义形式取决于服务器系统的产品的描述形式。

接口参考实例：见附录 C。

4.2.4 产品 BOM 提取接口。接口级别：二级

产品物料清单是一类在产品全生命周期中使用到的数据，对实现企业信息集成具有较大的作用。本接口提取产品的物料清单，按服务器给定的形式提供给客户程序。

接口参考实例：见附录 D。

4.2.5 设计过程数据提取接口。接口级别：二级

本接口使得客户程序可以按照服务器系统给定的格式提取过程数据。这些数据可以是任务属性、工作文档、评审意见等，数据的定义形式取决于服务器系统的描述形式。

接口参考实例：参见附录 E。

4.2.6 产品数据修改接口。接口级别：三级

本接口使得客户程序可以修改服务器系统中的产品数据。修改数据的规则由服务器接口定义。

接口参考实例：参见附录 F。

4.2.7 产品数据获取接口。接口级别：三级

本接口使得服务器可以从客户程序中获得所需的产品数据。即客户端程序通过调用本接口，可以将新的产品数据存入服务器系统，产品数据的格式由服务器接口定义。

接口参考实例：参见附录 F。

4.3 计算机辅助设计软件开放接口定义

CAD 的设计信息被广泛应用于 PDM、CAPP、CAM、ERP 等各种系统中。它是企业技术信息的重要的数据源。因此，开放 CAD 系统的相关数据接口是本规范中很重要的组成部分。本标准定义的 CAD 开放接口主要包括 CAD 系统中工程图纸的相关管理数据和通用图形格式转换接口。

4.3.1 CAD 图形浏览接口。接口级别：一级

本接口使得其他系统可以在 CAD 系统之外浏览相关 CAD 系统的图形文件。虽然现在已经推出了一些通用浏览器可以浏览一部分国际上较知名 CAD 系统的图形文件，但是 CAD 系统提供商开放其图形的浏览接口仍然是十分重要的，这是哪些通用浏览器具有长久生命力的重要保障。

接口参考实例：参见附录 G。

4.3.2 CAD 工程图管理信息提取接口。接口级别：二级

该接口主要用于提取 CAD 数据文件中的图纸管理信息，如图纸幅面、标题栏信息、明细表信息以及装配结构信息等。所提取的信息为 CAPP、PDM、ERP 等系统服务。

接口参考实例：参见附录 H。

4.3.3 CAD 图形数据转换接口。接口级别：二级

本接口提供了将一种 CAD 系统的图形文件转换为标准格式(例如 DXF、IGES 等)的接口。CAD 软件提供商通过该接口开放自己读入标准数据格式文件的功能以及输出为标准格式文件的功能。

接口参考实例：参见附录 J。

4.3.4 CAD 工程图管理信息修改接口。接口级别：三级

本接口要求 CAD 系统提供在其系统之外的修改 CAD 图形文件管理信息的接口。例如在将 CAD 设计文件存入 PDM 数据库以后，如果要在 PDM 中修改其标题栏、技术要求等有关管理方面的信息，则可以使用本接口。本接口使得这种修改可以实现信息变化的双向相关。

4.4 计算机辅助工艺设计软件开放接口定义

本条定义的接口主要涉及到产品加工工艺数据的存取，这部分数据主要由 CAPP 系统负责产生和管理。但是在有的系统中，工艺统计数据是由 PDM 系统产生和管理的，CAPP 系统只负责管理最原始的工艺卡片数据，并将其提交给 PDM 进行处理。也有一些 CAM 软件包含了 CAPP 软件的内容。总之无论这些工艺数据在哪些系统中定义，该系统都应按本标准开放相关的数据和功能的交换接口。

4.4.1 产品工艺数据浏览接口。接口级别：一级

本接口提供的是对工艺数据的浏览器。在不同的系统中，由于工艺数据的结构和体现形式不同，因而浏览方式不同。由于工艺数据的多样性，浏览器可以由一个接口给出，也可以由多个接口给出。例如在有的系统中，工艺卡片数据的浏览由 CAPP 负责，而工艺统计数据的浏览由 PDM 负责。各服务器可以根据其定义的工艺数据结构设计各自的浏览器。浏览器应可以浏览服务器中主要的工艺数据类型。

接口定义实例：参见附录 K 中的 IcappCardviewX 接口。

4.4.2 产品工艺数据提取接口。接口级别：二级

由于工艺数据的多样性，工艺数据的提取接口也可能有多个。本接口主要应满足 PDM 系统、ERP 系统、CAM 系统对工艺数据提取的需要。

接口参考实例：参见附录 K 中的 IprocessData 接口定义。

4.4.3 产品工艺统计数据提取接口。接口级别：二级

工艺统计数据也是工艺数据的一部分。但它们一般不是由工艺人员直接设计产生的，而是在系统对工艺卡片等直接设计的数据进行统计整理后产生的，主要为 ERP 系统提供生产准备数据。这部分数据在有的系统中由 PDM 负责提供。

接口参考实例：参见附录 K 中的 IcappSumData 接口定义。

5 符合性判定方法

为判定一种软件系统是否符合本标准的规定，应由规范的授权认证单位对相应软件进行符合性测

试。通过符合性测试的，可判定该软件系统在其符合性声明的范围内符合本标准。

5.1 符合性测试的目的

符合性测试的目的是按照规范所规定的接口范围和实现的功能，对被测软件产品进行测试，以判定其产品提供的集成接口是否达到规范所规定的要求。

5.2 符合性测试的套件

由于不同的软件产品所提供的用于符合性测试的中间件可能会采用不同的硬件、软件技术平台及中间件技术，所以测试的套件就会有较大差别。测试套件包括：测试对象、测试用例及测试环境。

5.2.1 测试对象

测试对象是由申请测试者提供的，它包括如下内容：

a) 提供符合本标准定义的开放接口的软件产品；

b) 和开放接口信息相关的数据组织结构，测试者可以据此判断开放的信息是否完整；

c) 符合本标准 4.1.2 的接口说明文档及规范实现的符合性声明，申请测试者在符合性声明中应给出测试对象符合本标准的范围，这个范围也是该软件的符合性测试的范围。

5.2.2 测试用例

申请测试者应提供可以公开的使用其接口的自测试用例及其源代码。测试用例的源代码可以用实现接口的中间件技术所支持的任何一种语言编写。

申请测试者还应提供使用测试用例进行的自测试的测试报告。

根据测试对象的符合性声明，需要时测试者可以在申请测试者提供的用例之外编写新的测试用例，以抽查测试对象在符合其中间件规范的其他环境中是否有效。

5.2.3 测试环境

申请测试者应明确声明可运行其测试对象的计算机软、硬件环境。

5.3 符合性测试的工作流程

本标准的符合性测试主要包括五个阶段：测试准备、静态测试、测试运行、结果分析、形成测试报告。

5.3.1 测试准备

在测试准备阶段测试者应编写测试计划。测试计划包括测试内容、测试环境、测试完成时应提供的测试文件、测试人员及其职责。

5.3.2 静态测试

静态测试主要是对测试对象所提供的文档进行审查，并确定测试进行的环境及测试用例。概括为以下几个方面：

a) 审查测试对象所提供的文档是否符合规范定义的格式。

b) 根据规范的定义和测试对象提供的符合性声明以及与开放接口信息相关的数据组织结构，判断测试对象实现接口的完整性。

c) 根据测试对象的符合性声明，在测试对象所提供的测试用例之外，测试者还应采用可以调用接口的其他语言编写测试用例以备测试。该用例应该由申请测试者认可，如果申请测试者不能认可该用例，则应将其理由记录备案，如果需要应在测试报告中对其进行说明。

5.3.3 测试运行

测试运行是运行可执行的测试套件的过程，这个过程要提供测试记录。测试记录应包括测试项目、执行描述、测试通过准则以及测试结果。

5.3.4 结果分析和测试报告

结果分析和测试报告是指按照规范的定义，对照测试记录进行差异性分析后，对各个测试项目是否符合本标准做出结论，结论可以是通过、不通过或无结论（对于无结论的测试必须详细阐述其理由）。然后形成测试报告对结论进行详细阐述。

附 录 A
（提示的附录）
产品数据浏览器接口实例

A1 接口定义

a）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下；

b）本接口描述采用 MIDL 格式，用 Microsoft Visual C++编写；

c）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等；

d）接口的 IDL 如下：

```
import "oaidl.idl";
import "ocidl.idl";

    [
        object,
        uuid(EE2E767E-AC08-11D4-9F99-00A00C138D91),
        dual,
        helpstring("IProductDataViewer Interface"),
        pointer_default(unique)
    ]
    interface IProductData Viewer:IDispatch
    {
        [id(1),helpstring("method ShowBOM")]HRESULT ShowBOM([in] BSTR ProdIDP,[in]
BSTR ProdVerP,[in]BSTR BOMType,[out,retval]LONG*statu);
        [id(2),helpstring("method ShowProductProperty")]HRESULT ShowProductproperty([in]
BSTR ProdIDP,[in]BSTR ProdVerP,[out,retval]LONG*statu);
        [id(3),helpstring("method ShowProductTech")]HRESULT ShowProductTech([in]BSTR
ProdIDP,[in]BSTR ProdVerP,[out,retval]LONG*statu);
        [id(4),helpstring("method ShowProductDocument")]HRESULT ShowProductDocument
([in]BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR DocType,[out,retval]LONG*statu);
        [id(5),helpstring("method ShowProductECO")]HRESULT ShowProductECO([in]BSTR
ProdIDP,[in]BSTR ProdVerP,[in]BSTR EcoID,[out,retval]LONG*statu);
};
```

A2 接口描述

本接口包含的方法如下：

A2.1 浏览产品结构

```
HRESULT ShowBOM([in]BSTR ProdIDP,//产品代号
    [in]BSTR ProdVerp,                    //产品版本号
    [in]BSTR BOMType,                  //产品结构类型
    [out,retval]LONG*statu                //返回值
```

);

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

产品版本号为空(" ")时,表示默认为产品的当前版本;产品结构类型为空(" ")时,表示默认为产品的设计 BOM。该方法先查询产品对应版本的相关结构类型,不存在则提示返回,存在则显示产品结构信息,(处理方法视不同 PDM 系统而定,可采用分层展开,即先展开产品层,根据交互来逐级展开其他层,也可采用完全展开,即一次展开所有层次),其中在完全展开方式中,遍历时需要增加产品结构循环嵌套(可能存在由于数据不规范,造成图号循环引用的情况)的判断处理,以避免死循环。

A2.2 浏览产品属性

```
HRESULT ShowProductProperty([in]BSTR ProdIDP,//产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

产品版本号为空(" ")时,表示默认为产品的当前版本。该方法用于显示产品相关的属性信息,对于不同的 PDM 系统所显示的内容不同,可以包括:代号、名称、设计者、负责人等设计属性。

A2.3 浏览产品技术要求

```
HRESULT ShowProductTech([in]BSTR ProdIDP,//产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

产品版本号为空(" ")时,表示默认为产品的当前版本。该方法用于显示产品相关的技术要求内容。

A2.4 浏览产品文档

```
HRESULT ShowProductDocument([in]BSTR ProdIDP,     //产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [in]BSTR DocType,                           //文档类型
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

产品版本号为空(" ")时,表示默认为产品的当前版本;文档类型为空(" ")时,表示默认为产品的主设计图(二维)或主模型文件(三维)。该方法先查询产品对应版本的相关文档类型,不存在则提示返回,存在则显示指定文档,完成从服务器中下载文档,并在浏览界面中打开(不同 PDM 系统的具体处理方法不一样)。

A2.5 浏览产品更改单

```
HRESULT ShowProductECO([in]BSTR ProdIDP,     //产品代号
    [in]BSTR ProdVerP,                          //产品版本号
    [in]BSTR EcoID,                             //更改单号
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

更改单号为空(" ")时,表示显示该版本产品对应的所有更改单,更改单显示内容由各 PDM 系统进行定义。

附 录 B
（提示的附录）
设计过程数据浏览器接口实例

B1 接口描述

a）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下；

b）本接口描述采用 MIDL 格式，用 Microsoft Visual C++编写；

c）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等；

d）接口的 IDL 如下：

```
interface IProjectData Viewer:IDispatch
{
    [id(1),helpstring("method ShowProjectStruct")]HRESULT ShowProjectStruct([in] BSTR PrjtID,[out,retval]LONG*statu);
    [id(2),helpstring("method ShowProjectProperty")]HRESULT ShowProjectProperty([in] BSTR PrjtID,[out,retval]LONG*statu);
    [id(3),helpstring("method ShowProjectDocument")]HRESULT ShowProjectDocument([in] BSTR PrjtID,[in] BSTR PrjtVer,[in] BSTR DocType,[out,retval]LONG*statu);
    [id(4),helpstring("method ShowAuditData")]HRESULT ShowAuditData([in] BSTR PrjtID,[in] BSTR PrjtVer,[out,retval]LONG*statu);
};
```

B2 接口描述

该接口包含的方法如下：

B2.1 浏览项目结构

```
HRESULT ShowProjectStruct([in]BSTR PrjtID,//项目代号
    [out,retval]LONG*statu                  //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

在不同 PDM 系统中对项目结构有不同的处理方法，视结构数据量的大小，可采用分层展开，即先展开产品层，根据交互来逐级展开其他层，也可采用完全展开，即一次展开所有层次。

B2.2 浏览任务属性

```
HRESULT ShowProjectProperty([in]BSTR PrjtID,    //项目代号
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该方法显示任务的属性，属性的具体内容由各 PDM 系统进行规定。

B2.3 浏览任务文档

```
HRESULT ShowProjectDocument([in]BSTR PrjtID,    //项目代号
    [in]BSTR PrjtVer,                           //项目版本号
    [in]BSTR DocType,                           //文档类型
```

```
    [out,retval]LONG* statu                    //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

项目版本号为空（“ ”）时，表示默认为项目的当前版本；文档类型为空（“ ”）时，显示该项目版本下的所有文档。该方法先查询产品对应版本的相关文档类型，不存在则提示返回，存在则显示指定文档，完成从服务器中下载文档，并在浏览界面中打开（不同 PDM 系统的具体处理方法不一样）。

B2.4 浏览任务评审数据

```
HRESULT ShowAuditData([in]BSTR PrjtID,     //项目代号
    [in]BSTR PrjtVer,                      //项目版本号
    [out,retval]LONG* statu                //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

项目版本号为空（“ ”）时，表示默认为项目的当前版本。该方法用于显示指定版本的项目（任务）相关的工作流程中的评审数据，具体内容视不同 PDM 系统而定。

附 录 C
（提示的附录）
产品数据提取接口实例

C1 接口定义

a）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下。

b）本接口描述采用 MIDL 格式，用 Microsoft Visual C++编写。

c）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

d）接口的 IDL 如下：

```
interface IProductDataContainer:IDispatch
{
    [id(1),helpstring("method GetProductProperty")]HRESULT GetProductProperty([in]BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR FieldType,[out]BSTR* FieldValue,[out,retval]LONG*statu);
    [id(2),helpstring("method GetProductTech")]HRESULT GetProductTech([in]BSTR ProdIDP,[in]BSTR ProdVerP,[out]BSTR*FileName,[out,retval]LONG*statu);
    [id(3),helpstring("method GetProductDocument")]HRESULT GetProductDocument([in]BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR FileType,[out]BSTR* FileName,[out,retval]LONG*statu);
    [id(4),helpstring("method GetProductECO")]HRESULT GetProductECO([in] BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR EcoNo,[out]BSTR*FileName,[out,retval]LONG*statu);
};
```

C2 接口描述

该接口包含的方法如下：

C2.1 获取产品属性

```
HRESULT GetProductProperty([in]BSTR ProdIDP,        //产品代号
    [in]BSTR ProdVerp,                               //产品版本号
    [in]BSTR FieldType,                              //属性字段
    [out]BSTR*FieldValue,                            //属性内容
    [out,retval]LONG*statu                           //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

其中属性字段用“^”来进行分隔组合，返回的属性内容与字段项相对应，也用“^”分隔，空内容用 NULL 来表示。

属性字段描述为：字段名称 1^字段名称 2^…^字段名称 n

属性内容描述为：字段内容 1^字段内容 2^…^字段内容 n

例如：

输入属性字段：“代号^名称^材料^图幅^图纸类型^设计者^设计日期^当前版本^……”

输出属性内容：“A000-101^车体支架^NULL^A1^装配图^陈刚^2000-3-2^2.0.1^……”

C2.2 获取产品技术要求

```
HRESULT GetProductTech([in]BSTR ProdIDP,        //产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [out]BSTR*FileName,                         //技术要求文件名
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

该接口调用后,返回一个 ASCII 码的文本文件,包含了对应版本产品的技术要求内容,具体格式由各 PDM 系统自定义。

C2.3 获取产品设计文档

```
HRESULT GetProductDocument([in]BSTR ProdIDP,    //产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [in]BSTR FileType,                          //文档说明
    [out]BSTR*FileName,                         //文档名称
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

该接口支持获取指定的设计文档,以及支持获取所有的设计文档,设计文档以下载到本地的方式,返回本地完整的文件名称(带路径)。

a) 获取指定的设计文档

文档说明用"^"来进行分隔组合,返回的文档名称与文档说明对应,也用"^"分隔。

例如:

输入文档类型:"设计图^测试结果^安装示意图^……"

输出文档名称:"C:\Output\A00-010.prt^C:\Output\report.doc^C:\Output\安装图.dwg^……"

b) 获取所有的设计文档

设置文档说明为空(""),此时,返回的文档名称在用"^"分隔的基础上,在每个文档名称后追加上对应的文档类型,用"$"来区分。

例如:

输入文档类型:""

输出文档名称:"C:\Output\A00-010.prt$设计图^C:\OutPut\report.doc$测试结果^C:\Output\安装图.dwg$安装示意图^……"

C2.4 获取产品产品更改通知单

```
HRESULT GetProductEco([in]BSTR ProdIDP,          //产品代号
    [in]BSTR ProdVerp,                          //产品版本号
    [in]BSTR EcoNo,                             //更改单编号
    [out]BSTR*FileName,                         //更改通知单文件名
    [out,retval]LONG*statu                      //返回值
);
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

该接口调用后,返回一个 ASCII 码的文本文件,包含了对应版本产品指定的更改单的内容,具体格式由各 PDM 系统自定义。如果更改单编号为空(""),将返回对应的所有更改单内容。

附 录 D
（提示的附录）
产品 BOM 提取接口实例

D1 接口描述

a）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下。

b）本接口描述采用 MIDL 格式，用 Microsoft Visual C++编写。

c）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

d）接口的 IDL 如下：

```
interface IProductBOMContainer:IDispatch
{
    [id(1),helpstring("method GetSimgleBOM")]HRESULT GetSimgleBOM([in]BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR BOMType,[in]BSTR FieldType,[out]BSTR*FileName,[out,retval]LONG*statu);
    [id(2),helpstring("method GetExpandBOM")]HRESULT GetExpandBOM([in]BSTR ProdIDP,[in]BSTR ProdVerP,[in]BSTR BOMType,[in]BSTR FieldType,[out]BSTR*FileName,[out,retval]LONG*statu);
};
```

D2 接口描述

该接口包含的方法如下：

D2.1 获取单层的产品 BOM

```
HRESULT GetSimgleBOM([in]BSTR ProdIDP,      //产品代号
    [in]BSTR ProdVerp,                      //产品版本号
    [in]BSTR BOMType,                       //BOM 类型
    [in]BSTR FieldType,                     //属性字段
    [out]BSTR*FileName,                     //返回的文件名
    [out,retval]LONG*statu                  //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该方法提供了获取指定版本的产品下一级装配结构信息。BOM 类型表示是设计 BOM、工艺 BOM、制造 BOM 等，具体由各 PDM 系统进行定义，如果为空时，默认为设计 BOM。属性字段是指获取的产品 BOM 的信息，各字段用“^”分隔组成。相应获取的值也以“^”分隔，其中空值用 NULL 表示，并存成 ASCII 码文件，把文件名返回程序调用者。

属性字段描述为：字段名称 1^字段名称 2^…^字段名称 n

属性内容描述为：字段内容 1^字段内容 2^…^字段内容 n

例如：

输入属性字段：“上层代号^上层图名^代号^名称^材料^图幅^图纸类型^设计者^设计日期^当前版本^……”

输出属性内容：“A000-101^车体支架^A000-101-1^横梁^NULL^A2^装配图^陈刚^2000-

3-2^2.0.1^……”

D2.2 获取展开的产品 BOM

```
HRESULT GetExpandBOM([in]BSTR ProdIDP,      //产品代号
    [in]BSTR ProdVerp,                      //产品版本号
    [in]BSTR BOMType,                       //BOM 类型
    [in]BSTR FieldType,                     //属性字段
    [out]BSTR*FileName,                     //返的文件名
    [out,retval]LONG*statu                  //返回值
);
```

该方法提供了获取指定版本的产品完整的装配结构信息，与“获取单层的产品 BOM**”方法的唯一不同是需要遍历展开产品的装配结构，再进行数据处理。参数说明及使用方法见** D2.1。

附 录 E
（提示的附录）
设计过程数据提取接口实例

E1 接口描述

a）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下。

b）本接口描述采用 MIDL 格式，用 Microsoft Visual C++编写。

c）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

d）接口的 IDL 如下：

```
interface IProjectDataContainer:IDispatch
{
    [id(1),helpstring("method GetProjectProperty")]HRESULT GetProjectProperty([in]
BSTR PrjtID,[in]BSTR FieldType,[out]BSTR*FieldValue,[out,retval]LONG*statu);
    [id(2),helpstring("method GetProjectDocument")]HRESULT GetProjectDocument([in]
BSTR PrjtID,[in]BSTR PrjtVer,[in]BSTR FileType,[out]BSTR*FileName,[out,retval]LONG*
statu);
    [id(3),helpstring("method GetAuditData")]HRESULT GetAuditData([in]BSTR PrjtID,
[in]BSTR PrjtVer,[out]BSTR*FileName,[out,retval]LONG*statu);
};
```

E2 接口描述

该接口包含的方法如下：

E2.1 获取任务属性

```
HRESULT GetProjectProperty([in]BSTR PrjtID,      //项目代号
    [in]BSTR FieldType,                          //属性字段
    [out]BSTR*FieldValue,                        //属性内容
    [out,retval]LONG*statu                       //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

其中属性字段用“^”来进行分隔组合，返回的属性内容与字段项相对应，也用“^”分隔，空内容用 NULL 来表示。

属性字段描述为：字段名称 1^字段名称 2^…^字段名称 n

属性内容描述为：字段内容 1^字段内容 2^…^字段内容 n

例如：

输入属性字段：“代号^名称^负责人^完成日期^状态^……”

输出属性内容：“P2000-10^GT 小车改进^陈刚^2000-7-20^评审……”

E2.2 获取任务文档

```
HRESULT GetProjectDocument([in]BSTR PrjtID,      //项目代号
    [in]BSTR PrjtVer,                            //项目版本号
    [in]BSTR FileType,                           //文档说明
```

```
    [out]BSTR*FileName,                    //文档名称
    [out,retval]LONG*statu                 //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该接口支持获取指定的任务文档，以及支持获取所有的任务文档，任务文档以下载到本地的方式，返回本地完整的文件名称（带路径）。项目版本号为空时，表示当前版本。

E2.3 获取指定的任务文档

文档说明用“^”来进行分隔组合，返回的文档名称与文档说明对应，也用“^”分隔。

例如：

输入文档类型“设计图^测试结果^安装示意图^……”

输出文档名称：“C:\Output\A00-010.prt^C:\OutPut\report.doc^C:\Output\安装图.dwg^……”

E2.4 获取所有的任务文档

设置文档说明为空（“”），此时，返回的文档名称在用“^”分隔的基础上，在每个文档名称后追加上对应的文档类型，用“$”来区分。

例如：

输入文档类型：“”

输出文档名称：“C:\Output\A00-010.prt$设计图^C:\Output\report.doc$测试结果^C:\Output\安装图.dwg$安装示意图^……”

E2.5 获取项目评审意见

```
HRESULT GetAuditData([in]BSTR PrjtID,      //项目代号
    [in]BSTR PrjtVer,                      //项目版本号
    [out]BSTR*FileName,                    //评审意见文件名
    [out,retval]LONG*statu                 //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该接口调用后，返回一个 ASCII 码的文本文件，包含了对应版本的任务相关的评审意见内容，具体格式由各 PDM 系统自定义。

附 录 F
（提示的附录）
产品数据修改接口实例

F1 接口定义

a）本接口用于从客户端获取产品数据及其结构定义的数据。通过新建功能可以从 PDM 外部建立产品的数据，通过更新功能可以从 PDM 外部修改产品的数据。产品数据中的产品结构用类似明细表的表达方式，将客户的装配图数据放入 PDM 数据库中，并自动建立产品的数据结构。接口可获取的产品数据包括产品的图纸、标题栏、技术要求。为了便于理解，图 F1 以下给出了本 PDM 中对产品数据的描述简图：

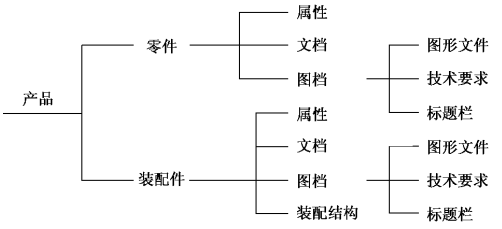

图 F1 PDM 开放数据组成

b）本接口运行在基于 MicroSoft Windows 操作系统的微机单机或网络环境下；

c）本接口描述采用 MIDL 格式，用 Visual Basic 语言编写；

d）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C＋＋、Visual Basic、VBA、Delphi、PowerBuilder 等；

e）接口的 IDL 如下：

```
[
  object,
  uuid(EBDC5E30-0092-11D4-AC87-0080C8761927),
  dual,
  helpstring("clsCad Interface"),
  pointer_default(unique)
]
interface clscad:IDispatch
        {
  [id(1),helpstring("method SaveBom")]HRESULT SaveBom([in]BSTR ProdIDP,[in]
BSTR ProdVerP,[in]BSTR FieldText,[in]BSTR Style,[in,optional]VARIANT Owner,[out,
retval]LONG*statu);
  [id(2),helpstring("method SaveTech")]HERESULT SaveTech([in]BSTR DrawNof,BSTR
DrawVerf,BSTR FileN,[out,retval]LONG*status);
  [id(3),helpstring("method SaveDwg")]HRESULT SaveDrawing(BSTR DrawNoP,BSTR
DrawVerP,BSTR FileN,BSTR ProdIDP,BSTR ProdVerP,[out,retval]LONG*status);
  [id(4),helpstring("method SaveTitle")]HRESULT SaveTitl(BSTR DrawNoP,BSTR
DrawVerP,BSTR FieldText,[out,retval]LONG*status);
```

};

f) 该接口在计算机中接口对象的名字为：PrjCadDll.ClsCad，它的服务器文件名为“Prjcaddll.dll”，支持 Automation 和 COM 双接口。

F2 接口描述

本接口包含的方法如下：

F2.1 产品明细入库

```
HRESULT SaveBom([in]BSTR ProdIDP,        //产品代号
    [in]BSTR ProdVerP,                   //产品版本号
    [in]BSTR FieldText,                  //明细表数据
    [in]BSTR style,                      //产品状态
    [in,optional]VARIANT Owner,          //字符型,产品的拥有者
    [out,retval]LONG*status                  //返回值
);
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

产品状态取值如下：“New”表示该产品为新入库，系统将删除原有数据再写入新的数据。如果存入的产品结构不存在，则本方法新建一个产品结构，同时建立产品结构上相关的产品基础属性数据。如果相应的产品结构已经存在，则刷新原有的数据。

明细表数据格式遵循如下规则：

a) 每一类明细属性及其数据用符号“^”分开，表示如下：

属性 1＝属性 1 值字符串^属性 2＝属性 2 值字符串……

b) 属性值的数据项用“&”分开，表示如下：

值 1&值 2&……

c) 各个属性值的项数必须相等；

d) 对于标准件必须提供规格属性；

e) 如果属性的某项没有值，则用“NULL”表示；

f) 对于单件重量用“$”将重量与单位分开，如 100$kg；

g) 对于标准件必须有规格，否则视同非标件；

h) 除了代号、非标准件版本号和标准件规格号以外，其他属性都可以省略；

i) 如果用户给出了系统未定义的属性，它将被忽略；

实例如下：序号＝10&11&1&2&3&4&5&6&7&8&9^代号＝GB 6174_86&GB 67_85&655-2.1.0&655D-1.0&655-2.1-0&655-2.2&655-2.3(1)&655-2.3(2)&655-2.4-0&655-2.5&655-2.1.5^名称＝六角薄螺母&开槽盘头螺钉&电池盒部&线路板(一)&壳体&堵盖&面板(一)&面板(二)&壳体档板&线路板立柱&触点簧片^数量＝6&2&1&1&1&2&1&1&1&4&2^材料＝HOb59-1&HPb59-1&NULL&NULL&ABS(桔红)&丁晴橡胶&NULL&PVC&ABS(桔红)&HPb59-1&3J9^单件＝NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL&NULL $NULL^总计＝NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL^备注＝NULL&镀银&NULL&NULL&NULL&NULL&选用国内&NULL&NULL&NULL&NULL^版本号＝NULL&NULL&1.0&1.0&1.0&1.0&1.0&1.0&1.0&1.0&1.0^规格＝M2&M2X6&NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL&NULL^标识＝标

准件 &.标准件 &.NULL&.NULL&.NULL&.NULL&.NULL&.选用德国 &.NULL&.NULL&.NULL。

F2.2 保存技术要求

```
HERESULT SaveTech([in]BSTR DrawNof,      //技术要求所属的图档代号
BSTR DrawVerf,                           //技术要求所属的图档版本号
BSTR FileN,                              //包含路径的技术要求文件名
[out,retval]LONG*status                  //返回值,成功时返回
                                         //VARIANT_TRUE,否则返回
                                         //VARIANT_FALSE
);
```

本接口创建或更新已有图档的技术要求。技术要求文件的格式只能采用纯 ASCII 码的形式,例如用 Windows NotePAD 编辑的文件。

F2.3 保存图档文件

```
HRESULT SaveDrawing(BSTR DrawNoP,        //图档代号
BSTR DrawVerP,                           //图档版本号
BSTR FileN,                              //带有路径的图档文件名
BSTR ProdIDP,                            //图档所属的产品代号
BSTR ProdVerP,                           //图档所属的版本号
[out,retval]LONG*status);                //返回值,成功时返回
                                         //VARIANT_TRUE,否则返回
                                         //VARIANT_FALSE
```

本方法将与产品有关的图档存入库中,一个产品可以有多个图档。如果相同的图档已经存在,则本方法更新图档的内容。

F2.4 保存图档的标题栏

```
HRESULT SaveTitl(BSTR DrawNoP,           //标题栏所属的图档代号
BSTR DrawVerP,                           //标题栏所属的图档版本号
BSTR FieldText,                          //标题栏字符串
[out,retval]LONG*status);                //返回值,成功时返回
                                         //VARIANT_TRUE,否则返回
                                         //VARIANT_FALSE
```

在调用本功能之前,标题栏必须在本 PDM 中已经定义。如果在标题栏字符串中出现的属性名称没有定义,则该属性被忽略。如果相应图档的标题栏已经存在,本方法更新标题栏内容。

附 录 G
（提示的附录）
CAD 系统开放接口实例

G1 CAD 系统开放接口模型（见图 G1）

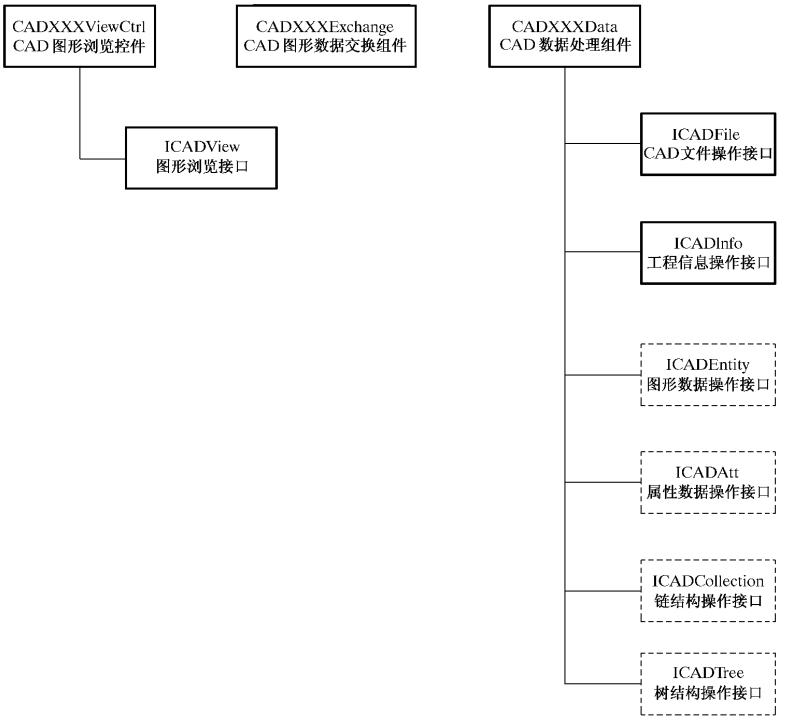

图 G1 CAD 系统开放模型

G2 CAD 图形浏览接口定义

a）本接口定义了一个 CAD 图形浏览控件 ICADView。通过该控件可以浏览 CAD 系统特定文件格式的图形，符合规范 4.3.1 的规定；

b）本接口运行在基于 MicroSoft Windows 操作系统的微机环境下；

c）本接口描述采用 ODL 格式，用 MFC 语言编写；

d）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等；

e）接口的 IDL 如下：

```
import "oaidl.idl";
import "ocidl.idl";
#include "olectl.h"
```

```
        object,
        uuid(7CC65900-900F-4EC5-9969-18464ACE7ABB),
        dual,
        helpstring("ICADView Interface"),
        pointer_default(unique)
    ]
    interface ICADView:IDispatch
    {
        [id(1),helpstring("method OpenFile")]HRESULT OpenFile([in]BSTR filename);
        [id(2),helpstring("method GetFileName")]HRESULT GetFileName([out]BSTR *
filename);
        [id(3),helpstring("method SetCommand")]HRESULT SetCommand([in]BSTR cmd-
str);
        [id(4),helpstring("method GetCommand")]HRESULT GetCommand([in]BSTR *
cmdstr);
    };
```

G3 CAD 图形浏览接口描述

G3.1 打开图形文件文件

```
HRESULT OpenFile(
    [in]BSTR filename                    //带路径的文件名
    );
```

方法调用成功时:返回 VARIANT_TRUE,**否则:**VARIANT_FALSE。

G3.2 获取当前打开的文件名

```
HRESULT GetFileName(
    [out]BSTR *filename                  //当前打开的带路径的文件名称
    );
```

方法调用成功时:返回 VARIANT_TRUE,**否则:**VARIANT_FALSE。

G3.3 发送命令

```
HRESULT SetCommand(
    [in]BSTR cmdstr                      //要设置的命令,有如下约定值
                                         //"pan"平移
                                         //"zoomwin"窗口放大
                                         //"zoomall"显示全部
                                         //"zoominout"动态缩放
                                         //"rotate"旋转变换(3D 独有)
    );
```

方法调用成功时:返回 VARIANT_TRUE,**否则:**VARIANT_FALSE。

G3.4 获取当前执行的命令

```
HRESULT GetCommand(
    [out]BSTR *cmdstr                    //当前命令字符串
    );
```

方法调用成功时:返回 VARIANT_TRUE,**否则:**VARIANT_FALSE。

附 录 H
（提示的附录）
CAD 工程图管理信息提取接口实例

H1 接口定义

a）本组件定义了一组用于对 CAD 图形信息处理的接口。它包括两个接口：ICADFile 及 ICADInfo。接口用于实现规范的 4.3.2 的规定。

b）本组件运行在基于 MicroSoft Windows 操作系统的微机环境下。

c）本组件描述采用 MIDL 格式，采用 Visual C++语言编写。

d）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

e）接口的 IDL 如下：

```
[
    object,
    uuid(4F6871BA-8B0B-4478-9D95-8630804714BC),
    dual,
    helpstring("ICADFile Interface"),
    pointer_default(unique)
]
interface ICADFile:IDispatch
{
    [id(1),helpstring("method OpenFile")]HRESULT OpenFile([in]BSTR filename);
    [id(2),helpstring("method GetFileName")]HRESULT GetFileName([out]BSTR *
filename);
    [id(3),helpstring("method GetVersion")]HRESULT GetVersion([out]BSTR *
version);
    [id(4),helpstring("method CloseFile")]HRESULT CloseFile();
};
[
    object,
    uuid(E7E8F5D0-5C66-487B-8722-37AA5A473F56),
    dual,
    helpstring("ICADInfo Interface"),
    pointer_default(unique)
]
interface ICADInfo:IDispatch
{
    [id(1),helpstring("method GetPaperSizeNo")]HRESULT GetPaperSizeNo([out]BYTE *
sizeno,[out]BYTE *direction);
    [id(2),helpstring("method GetPaperSize")]HRESULT GetPaperSize([out]double *width,
[out]double *height);
```

```
        [id(3),helpstring("method GetScaleFactor")]HRESULT GetScaleFactor([out]double *factor);
        [id(4),helpstring("method GetHeaderCount")]HRESULT GetHeaderCount([out]long *count);
        [id(5),helpstring("method GetHeaderItem")]HRESULT GetHeaderItem([in] VARIANT item,[out] BSTR *itemvalue);
        [id(6),helpstring("method GetBomCount")]HRESULT GetBomCount([out] long *count);
        [id(7),helpstring("method GetBomHeader")]HRESULT GetBomHeader([out] BSTR *header);
        [id(8),helpstring("method GetBomItem")]HRESULT GetBomItem ([in] int index,[out] BSTR *itemvalue);
        [id(9),helpstring("method GetInstruction")]HRESULT GetInstruction([out] BSTR *instruction);
        [id(10),helpstring("method GetModelType")]HRESULT GetModelType([out]BYTE *type);
        [id(11),helpstring("method GetModelName")]HRESULT GetModelName([out] BSTR *modelname);
        [id(12),helpstring("method GetAsmStructer")]HRESULT GetAsmStructer([out]BSTR *structure);
    };
```

H2 接口描述

H2.1 ICADFile 接口描述：

在使用 ICADData 组件的其他接口时，都必须先调用 ICADFile 接口方法打开 CAD 的数据文件。

H2.1.1 打开文件

```
HRESULT OpenFile(
    [in]BSTR filename                    //filename 要打开的带路径的文件名称
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.1.2 获取打开的文件名

```
HRESULT GetFileName(
    [out]BSTR *filename                  //filename 当前打开的文件名称(带路径)
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.1.3 获取打开的文件版本号

```
HRESULT GetVersion(
    [out]BSTR *version                   //version 为打开文件的版本号
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.1.4 关闭当前文件

```
HRESULT CloseFile();
```

关闭当前打开的文件，方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.2 ICADInfo 接口描述：

本接口提供的方法用于从 CAD 文件中获取用于管理方面的(如 PDM 存档)所需的数据。

H2.2.1 获取图幅代号

```
HRESULT GetPaperSizeNo(
    [out]BYTE *sizeno,              //sizeno:0、1、2、3、4、5 分别对应:A0、A1、A2、A3、A4、自定义
    [out]BYTE *direction            //direction:0—横放,1—竖放

    );
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE

H2.2.2 获取图幅尺寸

```
HRESULT GetPaperSize(
    [out]double *width,             //width:图纸宽
    [out]double *height             //height:图纸高
    );
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

H2.2.3 获取图幅绘制比例

```
HRESULT GetScaleFactor(
    [out]double *factor             //factor:绘图比例系数
    );
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

H2.2.4 获取标题栏属性数量

```
HRESULT GetHeaderCount(
    [out]long *count                //count:标题栏包含的项数,如(制图^ 张三)(日期^ 2000.10.10)
.....
    );
```

H2.2.5 获取标题栏属性值

```
HRESULT GetHeaderItem(
    [in]VARIANT item ,              //item:传入整型类型,按索引号返回项目,传入字符串型表示按
                                    //项目名称查找项目
    [out]BSTR *itemvalue            //itemvalue:按约定格式的字符串表示项目名称及项目值,
                                    //如"制图^ 张三"
    );
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

H2.2.6 获取明细表项目数

```
HRESULT GetBomCount(
    [out]long *count                //count:为图纸明细表项数(以行为单位)
    );
```

方法调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE。

H2.2.7 获取明细表头

```
HRESULT GetBomHeader(
    [out]BSTR *header               //图纸明细表表头信息字符串,如"序号^ 代号^ 名称^ 数量^
```

材料”

);

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.2.8 获取明细表项目值

```
HRESULT GetBomItem(
    [in]int index,                    //明细表项索引号，由下向上、由右到左
    [out]BSTR *itemvalue              //明细表项值，如“2^MDE-56-1-102^机盖^1^HT15-33”
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.2.9 获取技术要求文本

```
HRESULT GetInstruction(
    [out]BSTR *instruction            //技术要求文本字符串
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.2.10 获取模型类型

```
HRESULT GetModelType(
    [out]BYTE *type             //0—图纸，1—零件模型，2—装配模型
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

H2.2.11 获取模型名称

```
HRESULT GetModelName(
    [out]BSTR *modelname              //获得模型的名称
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该方法仅供零件模型和装配模型使用。

H2.2.12 获取装配描述符

```
HRESULT GetAsmStructer(
    [out]BSTR *structure              //装配结构描述字符串
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

该方法仅供装配模型使用。

例如：“0^锥齿轮减速箱
1^基座
1.1^基座主体
1.2^垫块
1.3^螺栓
2^圆锥大齿轮
3^圆锥小齿轮
4^调整垫片”

以上字符串条目之间用回车符分隔，描述了图 H1 所示装配结构。

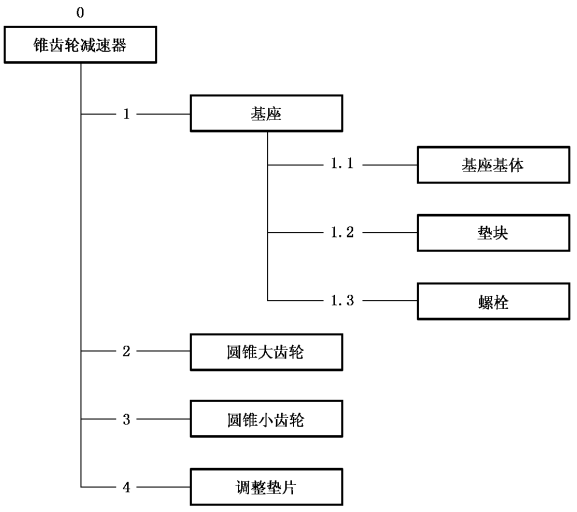

图 H1

附 录 J
（提示的附录）
CAD 图形数据转换接口实例

J1 ICADExchange 接口定义：

a）本接口定义了对 CAD 图形数据进行标准化转换的方法。该接口用于实现规范的 4.3.3 的规定。

b）本组件运行在基于 MicroSoft Windows 操作系统的微机环境下。

c）本组件描述采用 MIDL 格式，用 MFC 语言编写。

d）所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

e）接口的 IDL 如下：

```
//ICADExchange 组件对象的 idl 定义：
import "oaidl.idl";
import "ocidl.idl";
    [
        object,
        uuid(5D258D00-782F-4093-BA36-9550B74DB410),
        dual,
        helpstring("IExchange Interface"),
        pointer_default(unique)
    ]
    interface IExchange:IDispatch
    {
        [id(1),helpstring("method GetSupportIn")]HRESULT GetSupportIn([out]long *
count,[out]BSTR *exts,[out]BSTR *vers);
        [id(2),helpstring("method GetSupportOut")]HRESULT GetSupportOut([out]long *
count,[out]BSTR *exts,[out]BSTR *vers);
        [id(3),helpstring("method ReadFile")]HRESULT ReadFile([in]BSTR filename);
        [id(4),helpstring("method WriteFile")]HRESULT WriteFile([in]BSTR filename);
    };
[
    uuid(FF21666D-E9D1-49E8-9A47-144C0552626C),
    version(1.0),
    helpstring("CADExchange 1.0 Type Library")
]
library CADEXCHANGELib
{
    importlib("stdole32.tlb");
    importlib("stdole2.tlb");

    [
```

```
        uuid(E3A579F6-B393-4338-8096-770B1CCB4ACC),
        helpstring("Exchange Class")
    ]
    CoclassExchange
    {
        [default]interface IExchange;
    };
};
```

J2 ICADExchange 接口描述

J2.1 获取组件支持的输入文件格式

```
HRESULT GetSupportIn(
    [out]long *count,           //组件支持读入的文件种类数量
    [out]BSTR *exts,            //所支持的文件后缀字符串数组
    [out]BSTR *vers             //所支持的文件后缀类型版本号
    );
```

方法调用成功时：返回 VARIANT_TRUE，**否则**：VARIANT_FALSE。

J2.2 获取组件支持的输出文件格式

```
HRESULT GetSupportOut(
    [out]long *count,           //组件支持输出的文件种类数量
    [out]BSTR *exts,            //所支持的文件后缀字符串数组
    [out]BSTR *vers             //所支持的文件后缀类型版本号
    );
```

方法调用成功时：返回 VARIANT_TRUE，**否则**：VARIANT_FALSE。

J2.3 读入支持的格式文件

```
HRESULT ReadFile(
    [in]BSTR filename           //读入的文件名称(带路径)
    );
```

方法调用成功时：返回 VARIANT_TRUE，**否则**：VARIANT_FALSE。

J2.4 输出格式文件

```
HRESULT WriteFile(
    [in]BSTR filename           //输出的文件名称(带路径)
    );
```

方法调用成功时：返回 VARIANT_TRUE，**否则**：VARIANT_FALSE。

附 录 K
（提示的附录）
计算机辅助工艺设计软件开放接口实例

K1 CAPP 软件数据构成的接口对象结构关系模型（见图 K1）

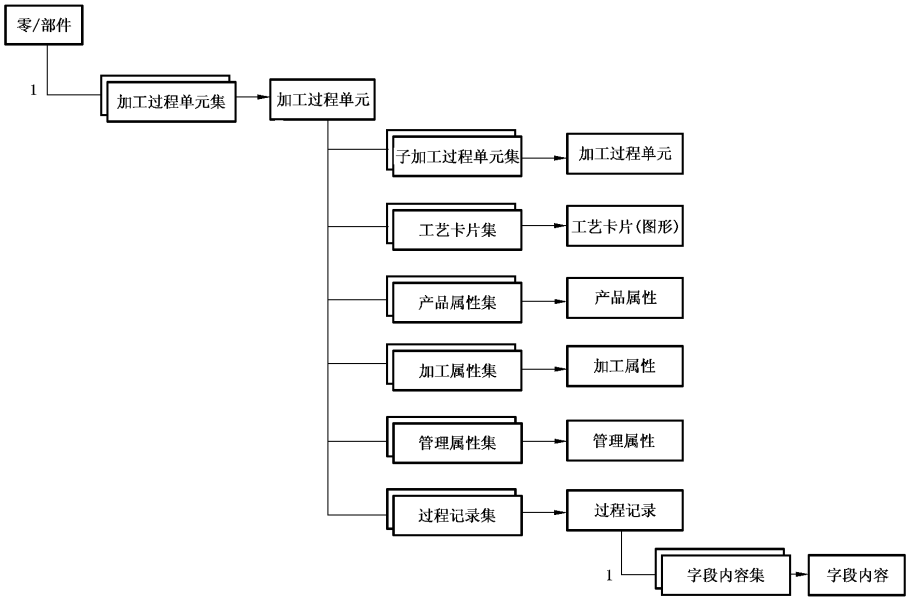

图 K1 CAPP 软件数据构成的接口对象结构关系模型

K2 CAPP 软件数据结构模型中接口对象的名称

零部件：IPart
加工过程单元集（加工过程单元）：IProcessUnits（IProcessUnit）
工艺卡片集（工艺卡片）：ICards（ICard）
产品属性集（产品属性）：IProductPropertys（IProductProperty）
加工属性集（加工属性）：IManufacturingPropertys（IManufacturingProperty）
管理属性集（管理属性）：IManagementPropertys（IManagementProperty）
过程记录集（过程记录）：IProcessRecordsets（IProcessRecordset）
字段内容集（字段内容）：IRecordFieldContents（IRecordFieldContent）

K3 CAPP 接口对象继承关系模型(见图 K2)

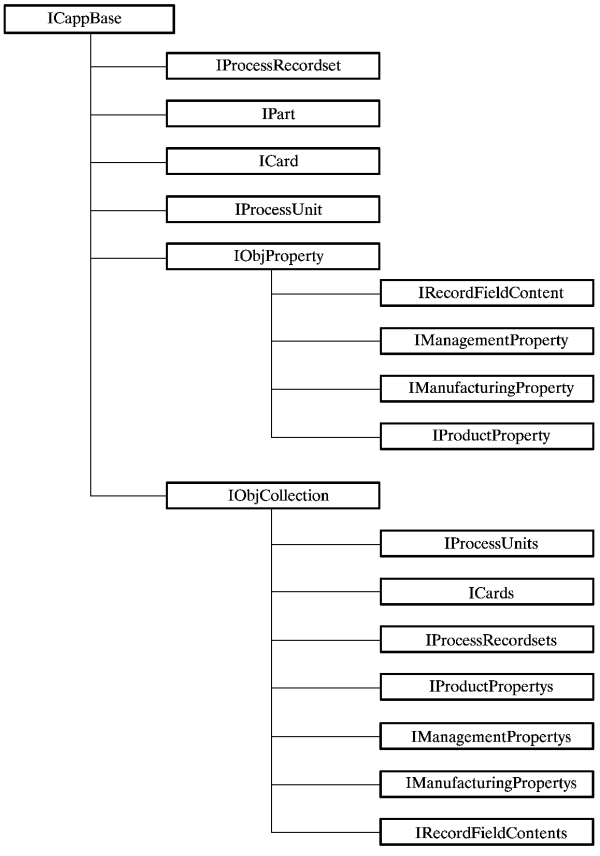

图 K2 CAPP 接口对象继承关系模型

在图 K2 中:

a) ICappBase:**所有接口对象类的基类**。

b) IObjProperty:**所有属性对象类的基类,它提供了访问所有属性对象的共同方法**。

c) IObjCollect:**所有集合对象类的基类,它提供了访问所有集合对象的共同方法**。

K4 CAPP 数据结构模型中接口对象定义

a) **本组件定义了一组用于** CAPP **软件系统开放数据存取的接口**。

b) **本组件运行在基于** MicroSoft Windows **操作系统的微机环境下**。

c) **本组件描述采用** MIDL **格式,用** MFC **语言编写**。

d) **所有支持** COM **技术的编程语言均可以使用本接口,例如** Visual C++、Visual Basic、VBA、Delphi、PowerBuilder **等**。

e) **接口的** IDL **如下**:

```
//icapp.idl:IDL source for icapp.dll

import "oaidl.idl";
import "ocidl.idl";
```

```
interface ICappBase:IDispatch
{
    [propget,id(1),helpstring("属性:对象的名称")]
        HRESULT Name([out,retval]BSTR *pVal);

    [id(2),helpstring("方法:得到最近的运行错误代码")]
        HRESULT GetLastErrorCode(int *iErrorCode);
};

interface IObjProperty:ICappBase
{
    [propget,id(1),helpstring("属性:属性名称")]
        HRESULT PropName([out,retval]BSTR *pVal);

    [propget,id(2),helpstring("属性:属性值")]
        HRESULT PropValue([out,retval]BSTR *pVal);
};

interface IObjCollection:ICappBase
{
    [propget,id(1),helpstring("属性:对象个数")]
        HRESULT ObjCount([out,retval]long *pVal);

    [id(2),helpstring("方法:通过顺序号得到集和中的对象")]
        HRESULT GetObjAt(long Index,ICappBase *pCappBase);

    [id(3),helpstring("方法:通过对象名称得到集和中的对象")]
        HRESULT GetObjByName(BSTR ObjName,ICappBase *pCappBase);
};

interface IParts:IObjCollection{};
interface IPart:ICappBase
{
    [propget,id(1),helpstring("属性:产品代号")]
        HRESULT ProductCode([out,retval]BSTR *pVal);

    [propget,id(2),helpstring("属性:产品的版本")]
        HRESULT ProductVersion([out,retval]BSTR *pVal);

    [propget,id(3),helpstring("属性:零部件代号")]
        HRESULT PartCode([out,retval]BSTR *pVal);
```

```
        [id(4),helpstring("方法:得到加工单元集")]
            HRESULT GetProcessUnits(IProcessUnits * pProcessUnits);

        [id(5),helpstring("方法:得到从属于该部件的零部件")]
            HRESULT GetSubParts(IParts * pParts);
    };

    interface IPartFactory:ICappBase
    {
        [id(1),helpstring("方法:创建 IPart 接口对象")]
            HRESULT Create(BSTR ProductCode,BSTR ProductVersion,BSTR PartCode,IPart *
pPart);
    };

    interface IProductProperty:IObjProperty{};
    interface IProductPropertys:IObjCollection{};

    interface IManagementProperty:IObjProperty{};
    interface IManagementPropertys:IObjCollection{};
    interface IManufacturingProperty:IObjProperty{};
    interface IManufacturingPropertys:IObjCollection{};

    interface IProcessUnits:IObjCollection{};
    interface IProcessUnit:ICappBase
    {
        [id(1),helpstring("方法:得到子加工单元集")]
            HRESULT GetSubUnits(IProcessUnits *pProcessUnits);

        [id(2),helpstring("方法:得到工艺卡片集")]
            HRESULT GetCards(ICards *pCards);

        [id(3),helpstring("方法:得到产品属性集")]
            HRESULT GetProductProps(IProductPropertys *pProdeuctPropertys);

        [id(4),helpstring("方法:得到加工属性集")]
                HRESULT GetManufacturingProps (IManufacturingPropertys * pManufac-
turingPropertys);

        [id(5),helpstring("方法:得到设计管理属性集")]
            HRESULT GetManagementProps(IManagementPropertys *pManagementPropertys);

        [id(6),helpstring("方法:得到过程记录集")]
            HRESULT GetProcessRecordsets(IProcessRecordsets *pProcessRecordsets);
```

```
};

interface ICards:IObjCollection{};
interface ICard:ICappBase
{
    [id(1),helpstring("方法:获得卡片图形文件")]
        HRESULT GetViewFile(BSTR FilePathAndName,BSTR *FileType);
};

interface IRecordFieldContent:IObjProperty{};
interface IRecordFieldContents:IObjCollection{};

interface IProcessRecordsets:IObjCollection{};
interface IProcessRecordset:ICappBase
{
    [id(1),helpstring("方法:得到当前的记录字段内容集")]
        HRESULT GetFieldContents(IRecordFieldContents *pFieldContents);

    [id(2),helpstring("方法:指向第一条记录")]
        HRESULT MoveFirst();

    [id(3),helpstring("方法:指向下一条记录")]
        HRESULT MoveNext();
};
```

K5 CAPP 数据结构模型中接口对象描述

K5.1 ICappBase 接口对象描述

```
interface ICappBase:IDispatch
{
    [propget,id(1),helpstring("属性:对象的名称")]
        HRESULT Name([out,retval]BSTR *pVal);

    [id(2),helpstring("方法:得到最近的运行错误代码")]
        HRESULT GetLastErrorCode(int *iErrorCode);
};
```

a) ICappBase 作为 CAPP 数据接口对象的基类来定义。

b) HRESULT Name([out,retval]BSTR *pVal)作为 ICappBase 接口的传出属性来定义,用于获得具体接口对象的名称。

c) HRESULT GetLastErrorCode(int *iErrorCode)作为 ICappBase 接口的一个方法来定义。实现该方法的目的是为了返回任何 CAPP 接口对象在执行过程中的出现错误的原因。iErrorCode 是该方法的一个传出参数。对于这个传出参数必须约定其意义。

例如：

```
#define IsOK                    0          //执行正确
#define NotImplement            -1         //不支持的操作
…
#define ErrorPara               -1000      //错误的参数
#define ParaOverScope           -1001      //参数超出范围
#define ErrorTypePara           -1002      //错误的参数类型…
```

在实际的设计工程软件的集成开发中，常常因为各种原因造成接口调用的失败，仅仅利用 TRUE 和 FALSE 来反映调用的状态，不利于更快速地找到接口调用失败的原因并解决出现的问题。所以建议在所有的 CAPP 接口对象的实现中尽量都对该方法给予支持。

这种约定是开放的、可以扩充的。其具体的含义可能与不同的系统对接口实现的方式有关。例如基于文件系统的 CAPP 与基于数据库系统的 CAPP 可能就不一样。

K5.2 IObjProperty 接口对象描述

```
interface IObjProperty:ICappBase
{
    [propget,id(1),helpstring("属性:属性名称")]
        HRESULT PropName([out,retval]BSTR *pVal);

    [propget,id(2),helpstring("属性:属性值")]
        HRESULT PropValue([out,retval]BSTR *pVal);
};
```

a) IObjeProperty 作为所有属性接口对象的基类来定义，用于获得某个属性的名称或属性值。

b) HRESULT PropName([out,retval]BSTR *pVal)作为 IObjProperty 接口中的传出属性定义，用于获得某属性对象的属性名称。

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

c) HRESULT PropValue([out,retval]BSTR * pVal)作为 IObjProperty 接口中的传出属性定义，用于获得某属性对象的属性值。

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

K5.3 IObjCollection 接口对象描述

```
interface IObjCollection:ICappBase
{
    [propget,id(1),helpstring("属性:对象个数")]
        HRESULT ObjCount([out,retval]long *pVal);

    [id(2),helpstring("方法:通过顺序号得到集和中的对象")]
        HRESULT GetObjAt(long Index,ICappBase *pCappBase);
    [id(3),helpstring("方法:通过对象名称得到集和中的对象")]
        HRESULT GetObjByName(BSTR ObjName,ICappBase *pCappBase);
};
```

a) IObjCollection 是所有 CAPP 集合接口对象的基类，用于实现对对象集合的管理，对象集合的遍历，以及对集合对象中某对象的查找。

b) HRESULT ObjCount([out,retval]long *pVal)作为 IObjCollection 接口的传出属性定义，用于获得对象集合中对象的个数。

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

c) HRESULT GetObjAt(long Index,ICappBase * pCappBase)作为 IObjCollection 接口的方法来定义，用于获得在对象集合中某顺序号的对象。

参数：

long Index　为指定的顺序号

ICappBase *pCappBase　为返回接口对象的指针

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

通过对 ObjCount(…)和 GetObjAt(…)的实现可以实现对对象集合的遍历，例如：

```
{
    long iObjCount;
    IObjCollection *pCollect=NULL;
    …//创建 pCollect 接口对象
    iObjCount=pCollect->GetObjCount();//注意属性的操作写法与 IDL 中有所不同。
    for(long iIndex=0;iIndex<iObjCount;iIndex++)
    {
    ICappBase *pCappBase=NULL;
    pCollect->GetObjAt(iIndex,pCappBase);
    …
}
```

1) HRESULT GetObjByName(BSTR ObjName,ICappBase * pCappBase)作为 IObjCollection 接口的方法来定义，用于在对象集合中查找已知对象名称的某对象，并返回该对象的接口指针。

参数：

BSTR ObjName　为指定对象的名称

ICappBase *pCappBase　为返回接口对象的指针

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

K5.4　IPart 接口对象描述

```
interface IParts:IObjCollection{};
interface IPart:ICappBase
{
    [propget,id(1),helpstring("属性:产品代号")]
        HRESULT ProductCode([out,retval]BSTR *pVal);

    [propget,id(2),helpstring("属性:产品的版本")]
```

```
        HRESULT ProductVersion([out,retval)] BSTR *pVal);
    [propget,id(3),helpstring("属性:零部件代号")]
        HRESULT PartCode([out,retval]BSTR *pVal);

    [id(4),helpstring("方法:得到加工单元集")]
        HRESULT GetProcessUnits(IProcessUnits *pProcessUnits);

    [id(5),helpstring("方法:得到从属于该部件的零部件")]
        HRESULT GetSubParts(IParts *pParts);
};

interface IPartFactory:ICappBase
{
    [id(1),helpstring("方法:创建 IPart 接口对象")]
        HRESULT Create(BSTR ProductCode,BSTR ProductVersion,BSTR PartCode,IPart *
pPart);
};
```

a) IPart 接口是所有其他 CAPP 数据接口的访问入口,因为从具体的工艺来考虑,工艺数据都是针对具体的零部件的。要获得某工艺数据首先要获得其针对的零部件。IPart 正是对零部件进行的接口定义。

其中最重要的方法是:HRESULT GetProcessUnits(IProcessUnits *pProcessUnits)用于得到该零部件的加工单元集合。参数是一个用于返回的 IProcessUnits 的接口对象的指针。

调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

b) IParts 接口是 IPart 的对象集合接口,主要用于在 CAPP 中实现对产品结构的描述,从而在工艺数据的汇总等操作中能实现更为实用的功能。

实现产品结构遍历的主要方法是 IPart 接口中的 HRESULT GetSubParts(IParts * pParts)方法,参数是 IParts 接口对象的指针,通过对 GetSubParts(…)的递归调用实现对产品结构的遍历。调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

c) IPartFactory 接口是用于创建 IPart 接口对象的接口。其中最重要的方法是

```
HRESULT Create (BSTR ProductCode,          //产品的代号
                BSTR ProductVersion,       //产品的版本
                BSTR PartCode,             //零部件代号
                IPart* pPart);             //用于返回的 IPart 接口对象指针
```

调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

K5.5 IProcessUnit 接口对象描述

```
interface IProcessUnits:IObjCollection{};
interface IProcessUnit:ICappBase
{
    [id(1),helpstring("方法:得到子加工单元集")]
```

```
        HRESULT GetSubUnits(IProcessUnits *pProcessUnits);

    [id(2),helpstring("方法:得到工艺卡片集")]
        HRESULT GetCards(ICards *pCards);

    [id(3),helpstring("方法:得到产品属性集")]
        HRESULT GetProductProps(IProductPropertys *pProductPropertys);

    [id(4),helpstring("方法:得到加工属性集")]
        HRESULT GetManufacturingProps(IManufacturingPropertys *pManufacturingPropertys);

    [id(5),helpstring("方法:得到设计管理属性集")]
        HRESULT GetManagementProps(IManagementPropertys *pManagementPropertys);

    [id(6),helpstring("方法:得到过程记录集")]
        HRESULT GetProcessRecordsets(IProcessRecordsets *pProcessRecordsets);
};
```

a) IProcessUnit 接口是 CAPP 接口对象模型中的重要接口。它实现了对各种工艺数据的管理和维护。

b) HRESULT GetSubUnits(IProcessUnits * pProcessUnits)作为 IProcessUnit 的一个方法来定义,用于获得从属于该工艺单元的子工艺单元集合。通过该方法的实现,可以遍历所有的工艺数据。

参数 IProcessUnits *pProcessUnits 是一个用于返回的 ProcessUnits 接口对象的指针。调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

c) HRESULT GetCards(ICards * pCards)作为 IProcessUnit 的一个方法来定义,用于获得该工艺单元的工艺卡片集合。

工艺卡片是在工艺设计的过程中产生的图形文件,该文件中包含文字、表格和图形,一般用于打印和浏览,是工艺数据的最终表现形式。

参数 ICards *pCards 是一个用于返回的 ICards 接口对象的指针,调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

d) HRESULT GetProductProps(IProductProperys * pProductPropertys)作为 IProcessUnit 的一个方法来定义,用于获得从属于该工艺单元的产品属性集合。

产品属性一般是在产品的设计过程中产生的,但在产品的工艺设计过程中要利用这些数据,在产品的工艺数据汇总中更要利用该数据,有些产品的属性还被要求填写在工艺卡片中,CAD 和 CAPP 的集成主要是该类数据的集成,所以该方法的实现是非常必要的。

参数 IProductProperys *pProductPropertys 是一个用于返回的 IproductPropertys 的对象指针。调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

e) HRESULT GetManufacturingProps(IManufacturingPropertys * pManufacturingPropertys)作为 IProcessUnit 的一个方法来定义,用于获得从属于该工艺单元的加工属性集合。

加工属性一般是在产品的工艺设计过程中产生的数据，一般要求填写在工艺卡片中，在产品的工艺数据汇总、CAPP 系统与 ERP 等系统的集成中一般都会用到该类数据。所以实现该方法是非常必备的。

参数 IManufacturingPropertys * pManufacturingPropertys 是一个用于返回的 IManufacturingPropertys 的对象指针，调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

f) HRESULT GetManagementProps (IManagementPropertys * pManagementPropertys) 作为 IProcessUnit 的一个方法来定义，用于获得从属于该工艺单元的管理属性集合。

管理属性一般是在产品的工艺设计过程中产生的数据，一般要求填写在工艺卡片中，管理属性的内容一般是该工艺的设计者、设计时间、修改时间、校核者、校核时间等内容。在 CAPP 系统与 PDM 等系统的集成中一般都会用到该类数据。所以实现该方法是必要的。

参数 IManagementPropertys * pManagementPropertys 是一个用于返回的 IManagementPropertys 的对象指针，调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

g) HRESULT GetProcessRecordsets (IProcessRecordsets * pProcessRecordsets) 作为 IProcessUnit 的一个方法来定义，用于获得从属于该工艺单元的加工过程记录集合。

加工过程记录数据一般是在产品的工艺设计过程中产生的数据，一般要求填写在工艺卡片中，是工艺设计的主要工作结果，大量的用于工艺的各种汇总中，在 CAPP 系统与 EPR 等系统的集成中一般都会用到该类数据。所以实现该方法是非常必要的。

参数 IProcessRecordsets * pProcessRecordsets 是一个用于返回的 IProcessRecordsets 的对象指针，调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

K5.6 ICard 接口对象描述

```
interface ICards:IObjCollection{};
interface ICard:ICappBase
{
    [id(1),helpstring("方法:获得卡片图形文件")]
        HRESULT GetViewFile(BSTR FilePathAndName,BSTR *FileType);
};
```

a) ICard 接口是对获得工艺卡片而定义的，工艺卡片是在工艺设计的过程中产生的图形文件，该文件中包含文字、表格和图形，一般用于打印和浏览，是工艺数据的最终表现形式。

b) HRESULT GetViewFile(BSTR FilePathAndName,BSTR * FileType)作为 ICard 接口中的一个方法来定义的，用于获得工艺卡片的图形文件。

参数：

BSTR FilePathAndName 作为传入参数，用于指定生成工艺卡片图形文件的路径和名称。BSTR * FileType 作为传入参数，用于指定生成工艺卡片图形文件的格式，如果 FileType 为 NULL，则该方法用缺省的图形文件的格式来生成工艺卡片。

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

K5.7 IProcessRecordset 接口对象描述

```
interface IRecordFieldContent:IObjProperty{};
interface IRecordFieldContents:IObjCollection{};
```

```
interface IProcessRecordsets:IObjCollection{};
interface IProcessRecordset:ICappBase
{
    [id(1),helpstring("方法:得到当前的记录字段内容集")]
        HRESULT GetFieldContents(IRecordFieldContents *pFieldContents);

    [id(2),helpstring("方法:指向第一条记录")]
        HRESULT MoveFirst();

    [id(3),helpstring("方法:指向下一条记录")]
        HRESULT MoveNext();

};
```

a) IProcessRecordset 接口是对加工过程记录数据设计的,用于管理维护和访问加工过程记录数据。

加工过程记录数据一般是在产品的工艺设计过程中产生的数据,一般要求填写在工艺卡片中,是工艺设计的主要工作结果,大量的用于工艺的各种汇总中,在 CAPP 系统与 EPR 等系统的集成中一般都会用到该类数据。

b) HRESULT GetFieldContents(IRecordFieldContents * pFieldContents)作为 IProcessRecordset 接口的一个方法来定义,用于获得当前记录的字段内容集合。

参数 IRecordFieldContents *pFieldContents 是一个用于传出的 IRecordFieldContents 接口对象指针,通过对 IRecordFieldContents 接口对象的操作,可以得到记录字段的个数,每个字段的名称,每个字段的内容等。

调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

c) HRESULT MoveFirst()和 HRESULT MoveNext()作为 IProcessRecordset 接口对象的方法来实现,主要用于改变当前的工艺过程记录。

IProcessRecordset 接口通过 MoveFirst()以及 MoveNext()接口方法实现工艺过程记录的遍历,如果 GetFieldContents 方法中返回的 pFieldContents 为 NULL 表示已经遍历完毕。

调用成功时:返回 VARIANT_TRUE,否则:VARIANT_FALSE;如果要进一步获得调用失败的原因,可调用父类接口的 GetLastErrorCode(…)方法来实现。

K6 CAPP 工艺卡片浏览接口对象定义

a) 本组件定义了一个用于浏览 CAPP 工艺卡片的接口。

b) 本组件运行在基于 MicroSoft Windows 操作系统的微机环境下。

c) 本组件描述采用 MIDL 格式,用 MFC 语言编写。

d) 所有支持 COM 技术的编程语言均可以使用本接口,例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

e) 接口的 IDL 如下:

```
interface ICappCardViewX:IDispatch
{
```

```
//浏览窗口的句柄,传出属性
    [id(0),propget helpstring("浏览窗口的句柄,传出")]
        HRESULT Window([out,retval]long *phwnd);

//浏览文件源的路径和文件名,传入传出属性
    [id(1),propget, helpstring("浏览文件源的路径和文件名,传出")]
        HRESULT src([out,retval]BSTR *pVal);
    [id(1),propput, helpstring("浏览文件源的路径和文件名,传入")]
        HRESULT src([in]BSTR *pVal);
//当前的浏览方式,传入传出属性
    [id(2),propget, helpstring("得到当前的浏览方式,传出,如 Pan,Zoom,ZoomToRect,…")]
        HRESULT UserMode([out,retval]BSTR *pVal);
    [id(2),propput, helpstring("改变当前的浏览方式,传入,如 Pan,Zoom,ZoomToRect,…")]
        HRESULT UserMode([in]BSTR *pVal);

//支持路径,传入传出属性
    [id(3),propget, helpstring("支持路径,传出")]
        HRESULT SupportPath([out,retval]BSTR *pVal);
    [id(3),propput, helpstring("支持路径,传入")]
        HRESULT SupportPath([in]BSTR *pVal);

//字体路径,传入传出属性
    [id(4),propget, helpstring("字体路径,传出")]
        HRESULT FontPath([out,retval]BSTR *pVal);
    [id(4),propput, helpstring("字体路径,传入")]
        HRESULT FontPath([in]BSTR *pVal);
//当前浏览坐标范围,传入传出属性
    [id(5),propget, helpstring("当前浏览范围")]
        HRESULT CurrentView(
            [out] double* left,
            [out] double* right,
            [out] double* bottom,
            [out] double* top);

    [id(5),propput, helpstring("当前浏览范围")]
        HRESULT CurrentView(
            [in] double left,
            [in] double right,
            [in] double bottom,
            [in] double top);
//整个卡片的坐标范围,传出属性,主要用于全部显示计算图形范围
    [id(5),propput, helpstring("整个卡片的坐标范围")]
        HRESULT GetDrawingExtents(
```

```
        [out] double* left,
        [out] double* right,
        [out] double* bottom,
        [out] double* top);
};
```

K7 CAPP 工艺卡片浏览接口对象描述

本接口用于实现对工艺卡片的浏览，包含的方法如下：

K7.1 获得浏览窗口的句柄

```
HRESULT Window([out,retval]long *phwnd//窗口句柄
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

K7.2 指定浏览文件源的路径和文件名

```
HRESULT src([in]BSTR *pVal          //指定的文件源路径和名称
    );
```

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

K7.3 设置当前浏览的工作方式

```
HRESULT UserMode([in]BSTR *pVal          //浏览的工作方式
    );
```

对浏览的工作方式用字符串来表达，同时对字符串对应的操作有一定的约定，例如：

"Pan"对应平移操作，"Zoom+"对应放大操作，"Zoom-"对应缩小操作，"ZoomToRect"对应缩放到用户指定的窗口操作等。

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

K7.4 设置当前浏览的坐标范围

```
HRESULT CurrentView(
        [in] double left,        //左下角 x 坐标
        [in] double right,       //右上角 x 坐标
        [in] double bottom,      //左下角 y 坐标
        [in] double top);        //右上角 y 坐标
        );
```

用于通过程序而不是用户交互来设置当前浏览的坐标范围。

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

K7.5 获得整个卡片的坐标范围

```
HRESULT GetDrawingExtents(
        [out] double* left,      //左下角 x 坐标
        [out] double* right,     //右上角 x 坐标
        [out] double* bottom,    //左下角 y 坐标
        [out] double* top);      //右上角 y 坐标
        );
```

用于获取整个卡片的图形浏览范围。

方法调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE。

K8 CAPP 工艺过程数据提取接口对象定义

a) 本组件定义了一个用于工艺过程数据提取的接口。

b) 本组件运行在基于 MicroSoft Windows 操作系统的微机环境下。

c) 本组件描述采用 MIDL 格式,用 MFC 语言编写。

d) 所有支持 COM 技术的编程语言均可以使用本接口,例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

e) 接口的 IDL 如下:

```
interface IProcessData:ICappBase
{
    [id(1),helpstring("方法:获得产品工艺过程数据")]
    HRESULT GetData([in]BSTR ProductCode,          //产品代号
                    [in]BSTR ProductVersion,       //产品版本
                    [in]BSTR PartCode,             //零部件代号
                    [in]BSTR ProcessName,          //过程名称
                    [in]BSTR ProcessDataFile);     //过程数据文件名称
};
```

K9 CAPP 工艺过程数据提取接口对象描述

本接口用于获取产品工艺过程数据。

通过 CAPP 接口对象模型中提供的接口对象的组合使用,完全可以获得产品工艺过程的所有数据。为了简化该数据的获得,IProcessData 接口提供一个直接获得产品工艺过程数据的接口对象的定义。

K9.1 获取过程数据

```
HRESULT GetData([in]BSTR ProductCode,          //产品代号
                [in]BSTR ProductVersion,       //产品版本
                [in]BSTR PartCode,             //零部件代号
                [in]BSTR ProcessName,          //过程名称
                [in]BSTR ProcessDataFile);     //过程数据文件名称
```

使用该方法需要提供的参数是产品的代号、产品的版本、零部件代号、工艺过程名称。获得的数据存放在参数 ProcessDataFile 指定的文件中。

工艺过程数据文件的可以参照以下协议生成:

a) 字段名称以及字段内容之间用符号"^ "分开,表示如下:

字段名称 1^ 字段名称 2^ …^ 字段名称 n

字段内容 1^ 字段内容 2^ …^ 字段内容 n

b) 数据的每行数据之后都有回车换行符号;

c) 数据的第一行是关于过程数据的字段定义,以后每行数据对应过程数据的内容;

d) 过程内容以 NULL 结尾。

例如:

工序号^ 工序名称^ 车间^ 设备名称型号^ 工装名称^ 工装代号^ 单件工时

0^ 备料,热轧圆钢 ϕ38×678^ 库备^ ^ ^ ^

…

80^ 磨外圆至 ϕ32-0.12-0.28 处^ 二车间^ 外圆磨床 M131^ ^ ^

调用成功时：返回 VARIANT_TRUE，否则：VARIANT_FALSE；如果要进一步获得调用失败的原因，可调用父类接口的 GetLastErrorCode(…)方法来实现。

K10 CAPP 工艺汇总数据提取接口对象定义

a) 本组件定义了一个用于工艺汇总数据提取的接口。

b) 本组件运行在基于 MicroSoft Windows 操作系统的微机环境下。

c) 本组件描述采用 MIDL 格式，用 MFC 语言编写。

d) 所有支持 COM 技术的编程语言均可以使用本接口，例如 Visual C++、Visual Basic、VBA、Delphi、PowerBuilder 等。

e) 接口的 IDL 如下：

```
interface ICappSumData:ICappBase
{
    [id(1),helpstring("方法:获得产品工艺统计数据")]
    HRESULT GetData([in]BSTR ProductCode,          //产品代号
                    [in]BSTR ProductVersion,       //产品版本
                    [in]BSTR PartCode,             //零部件代号
                    [in]BSTR DataSrcDefine,        //提取数据源字段定义
                    [in]BSTR SumDataFile);         //提取结果数据文件名称
};
```

K11 CAPP 工艺汇总数据提取接口对象描述

本接口提供了工艺数据汇总的功能。产品工艺统计数据中涉及的数据非常多，有产品的属性数据，产品的加工属性数据，以及产品的加工过程记录数据等。同时每个企业中的工艺汇总所需要的数据又是不一样的，可以通过 CAPP 接口对象模型中提供的接口对象的组合使用，来获得产品工艺过程的所有数据。为了简化该数据的获得，本规范提供一个直接获得产品工艺统计数据的数据提取接口对象的定义。数据提取之后的统计、合并、汇总工作由其他系统解决。

数据汇总的对象是某个产品或某个零部件下的所有零部件，汇总的过程中需要遍历产品结构。

K11.1 工艺数据汇总

```
    HRESULT GetData([in]BSTR ProductCode,          //产品代号
                    [in]BSTR ProductVersion,       //产品版本
                    [in]BSTR PartCode,             //零部件代号
                    [in]BSTR DataSrcDefine,        //提取数据源字段定义
                    [in]BSTR SumDataFile);         //提取结果数据文件名称
```

a) 工艺数据汇总接口方法 GetData(…)需要的基本参数是产品的代号、产品的版本、零部件的代号。

b) 需要获取的数据源定义，是 BSTR DataSrcDefine 参数，该方法通过该参数的定义描述来提取数据，如果该参数为 NULL，则不提取任何数据。

有关汇总数据源字段定义，可以参照以下解释协议。

DataSrcDefine 是由多行字符串组成的一个字符串，行与行之间用回车换行符分隔。DataSrcDefine 以 NULL 结尾。

每行数据定义的是汇总结果中的一个字段内容定义。其表示方法为：

字段名称：加工单元名称 1|数据集合名称|属性或字段名称^ 加工单元名称 2|数据集合名称|属性或工艺记录字段名称

字段名称和字段定义之间用“：”分隔，加工单元之间通过“^ ”分隔，名称之间通过“|”分隔；工艺记录字段名称用“工艺记录名称～记录字段名称”表示。

1) 加工单元名称是指 IProcessUnit 接口对象的名称；

2) 数据集合名称有三个，分别是“产品属性”、“加工属性”、“过程记录”；

3) 属性或字段名称是指 IProductProperty、IManufacturingProperty、IManagementProperty 接口对象名称或是 IProcessRecordset～IRecordFieldContent 名称的接口对象组合名称(IProcessRecordset～IRecordFieldContent)反映的是某过程记录集中的某字段)。

例如：需要对某产品加工过程中所有的材料数据进行提取，对材料名称的提取定义如下：

材料名称：锻造工艺|加工属性|材料牌号^ 机加工工艺|加工属性|材料牌号^ 油漆涂覆工艺|过程记录～工序|涂覆材料。

以上的定义表示：提取后的字段名称为材料名称，它从锻造工艺、机加工工艺和油漆涂覆工艺三个加工单元中提取；对于锻造工艺，提取的是加工属性中的材料牌号；对于机加工工艺提取的也是加工属性中的材料牌号；而对油漆涂覆工艺中提取的是工序过程记录中每条记录中的涂覆材料字段的内容。

c) 获得的获得的数据存放在参数 SumDataFile 指定的文件中。

文件可以参照以下协议生成：

字段名称以及字段内容之间用符号“^ ”分开，表示如下：

字段名称 1^ 字段名称 2^ …^ 字段名称 n

字段内容 1^ 字段内容 2^ …^ 字段内容 n

d) 数据的每行数据之后都有回车换行符号。

e) 数据的第一行是关于过程数据的字段定义，以后每行数据对应过程数据的内容。

f) 过程内容以 NULL 结尾。

中 华 人 民 共 和 国
国 家 标 准
现代设计工程集成技术的软件
接 口 规 范
GB/T 18726—2002
*
中国标准出版社出版
北京复兴门外三里河北街 16 号
邮政编码：100045
电话：68523946 68517548
中国标准出版社秦皇岛印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
开本 880×1230 1/16 印张 3¼ 字数 92 千字
2002 年 12 月第一版 2002 年 12 月第一次印刷
印数 1—1 500
*
书号：155066·1-19036 定价 21.00 元
网址 www.bzcbs.com
*
科 目 629—500

GB/T 18726-2002