[19] 中华人民共和国国家知识产权局

[51] Int. Cl[7]
G06F 9/44

[12] 发明专利申请公开说明书

[21] 申请号 200410078529.1

[43] 公开日 2005 年 4 月 27 日

[11] 公开号 CN 1609794A

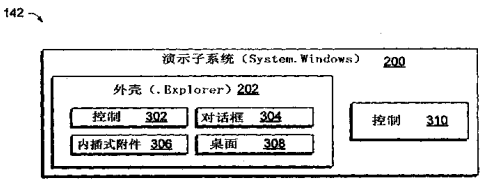
[22] 申请日 2004.9.10
[21] 申请号 200410078529.1
[30] 优先权
[32] 2003.10.24 [33] US [31] 10/693,718
[71] 申请人 微软公司
地址 美国华盛顿州
[72] 发明人 C·J·古扎克 K·B·卡拉塔尔
M·M·米勒 M·G·谢尔顿
T·P·麦克基
[74] 专利代理机构 上海专利商标事务所有限公司
代理人 张政权

权利要求书 7 页 说明书 38 页 附图 9 页

[54] 发明名称 用于计算机平台的编程接口

[57] 摘要

一种用于计算机平台的编程接口可包括各种功能。在某些实施例中，该编程接口包括与可重复使用的用户界面控制有关的第一组服务、与用户界面对话框和用户界面向导有关的第二组服务、与扩展用户界面功能有关的第三组服务以及与扩展用户界面的桌面的功能有关的第四组服务的一个或多个。



ISSN1008-4274

1．一种收录在一个或多个计算机可读媒质上的编程接口，其特征在于，包括：

与可重复使用的用户界面控制有关的第一组服务；

与用户界面对话框和用户界面向导有关的第二组服务；

与扩展所述用户界面功能有关的第三组服务；以及

与扩展所述用户界面的桌面的功能有关的第四组服务。

2．如权利要求1所述的编程接口，其特征在于，所述第一组服务包括：

封装存储用户经验的第一控制；

允许以应用定义的方式显示项目的第二控制；

允许向所述桌面的边条添加项目的第三控制；

允许向列表添加项目的第四控制；以及

允许显示项目的预览图像的第五控制。

3．如权利要求1所述的编程接口，其特征在于，所述第二组服务包括：

允许打开和保存文件和文件夹的第一对话框；

允许向光盘进行写入的第一向导；以及

方便通过电子邮件发送图像的第二向导。

4．如权利要求1所述的编程接口，其特征在于，所述第三组服务包括：

允许向上下文菜单进行添加的第一功能；

允许标识应用定义的缩略图的第二功能；以及

允许当显示关于一个或多个文件或文件夹的信息时执行计算的第三功能。

5．如权利要求1所述的编程接口，其特征在于，所述第四组服务包括：

允许在所述桌面上显示边条的第一功能；以及

允许在所述桌面上显示应用定义的通知的第二功能。

6．一种系统，其特征在于，包括：

用于展现启用用户界面的可重复使用的控制的第一组功能的装置；

用于展现启用所述用户界面的可重复使用的对话框和所述用户界面的可重复使用的向导的第二组功能的装置；以及

用于展现启用所述用户界面的桌面的功能扩展的第三组功能的装置。

7．如权利要求6所述的系统，其特征在于，还包括：

用于展现启用所述用户界面的功能扩展的第四组功能的装置。

8. 如权利要求 7 所述的系统，其特征在于，用于展现第四组功能的装置包括：

用于展现允许向上下文菜单进行添加的一个或多个功能的装置；

用于展现允许标识应用定义的缩略图的一个或多个功能的装置；以及

用于展现允许在显示关于一个或多个文件或文件夹的信息时执行计算的一个或多个功能的装置。

9. 如权利要求 6 所述的系统，其特征在于，用于展现第一组功能的装置包括：

用于展现封装存储用户经验的一个或多个功能的装置；

用于展现允许以应用定义的方式显示项目的一个或多个功能的装置；

用于展现允许向所述桌面的边条添加项目的一个或多个功能的装置；

用于展现允许向列表添加项目的一个或多个功能的装置；以及

用于展现允许显示项目的预览视图的一个或多个功能的装置。

10. 如权利要求 6 所述的系统，其特征在于，用于展现第二组功能的装置包括：

用于展现允许打开和保存文件和文件夹的一个或多个功能的装置；

用于展现允许向光盘进行写入的一个或多个功能的装置；以及

用于展现允许重新调整通过电子邮件发送的图像的大小的一个或多个功能的装置。

11. 如权利要求 6 所述的系统，其特征在于，用于展现第三组功能的装置包括：

用于展现允许在所述桌面上显示边条的一个或多个功能的装置；以及

用于展现允许在所述桌面上显示应用定义的通知的一个或多个功能的装置。

12. 一种将用户界面的一组类型组织成分层名字空间的方法，其特征在于，包括：

从所述类型组创建多个组，每一组包含展现逻辑上相关的功能的至少一种类型；

向所述多个组的每一组分配名字，其中，所述多个组中的一个组包括与可重复使用的用户界面控制有关的功能，所述多个组中的另一组包括与可重复使用的用户界面对话框和可重复使用的用户界面向导有关的功能；以及

选择一顶层标识符并用所述顶层标识符对每一组的名字加上前缀，使得每一

组中的类型由一分层名字来引用，所述分层名字包括作为包含所述类型的所述组的名字的前缀的所选择的顶层标识符。

13．如权利要求 12 所述的方法，其特征在于，所述多个组中的还有一组包括与扩展用户界面功能有关的功能。

14．如权利要求 13 所述的方法，其特征在于，与扩展用户界面功能有关的功能包括：

允许向上下文菜单进行添加的第一功能；

允许缩略图的应用标识的第二功能；以及

允许当显示关于一个或多个文件或文件夹的信息时执行计算的第三功能。

15．如权利要求 13 所述的方法，其特征在于，所述分配包括向包括与扩展用户界面功能有关的功能的组分配名字 Addin，使得包括与扩展用户界面功能有关的功能的组的分层名字为 System.Windows.Explorer.Addin。

16．如权利要求 12 所述的方法，其特征在于，所述多个组的还有一组包括与扩展用户界面桌面的功能有关的功能。

17．如权利要求 16 所述的方法，其特征在于，与扩展用户界面桌面的功能有关的功能包括：

允许在所述桌面上显示边条的第一功能；以及

允许在所述桌面上显示应用定义的通知的第二功能。

18．如权利要求 16 所述的方法，其特征在于，所述分配包括向包括与扩展用户界面桌面的功能有关的功能的组分配名字 Desktop，使得包括与扩展用户界面桌面的功能有关的功能的组的分层名字为 System.Windows.Explorer.Desktop。

19．如权利要求 12 所述的方法，其特征在于，所述分配包括向包括与可重复使用的用户界面控制有关的功能的组分配名字 Controls，使得包括与可重复使用的用户界面控制有关的功能的组的分层名字为 System.Windows.Explorer.Controls，并且向包括与可重复使用的用户界面对话框和可重复使用的用户界面向导有关的功能的组分配名字 Dialogs，使得包括与可重复使用的用户界面对话框和可重复使用的用户界面向导有关的功能的组的分层名字为 System.Windows.Explorer.Dialogs。

20．如权利要求 12 所述的方法，其特征在于，与可重复使用的用户界面控制有关的功能包括：

封装存储用户经验的第一控制；

允许以应用定义的方式显示项目的第二控制；

允许向列表添加项目的第三控制；以及

允许显示项目的预览图像的第四控制。

21．如权利要求 12 所述的方法，其特征在于，与可重复使用的用户界面对话框和可重复使用的用户界面向导有关的功能包括：

允许打开和保存文件和文件夹的第一对话框；

允许向光盘进行写入的第一向导；以及

方便通过电子邮件发送图像的第二向导。

22．一种方法，其特征在于，包括：

创建具有启用用户界面的功能扩展的功能的第一名字空间；以及

创建具有启用所述用户界面的桌面的功能扩展的功能的第二名字空间。

23．如权利要求 22 所述的方法，其特征在于，还包括：

创建具有启用可重复使用的用户界面控制的功能的第三名字空间。

24．如权利要求 23 所述的方法，其特征在于，所述第三名字空间包括：

封装存储用户经验的第一控制；

允许以应用定义的方式显示项目的第二控制；

允许向所述桌面的边条添加项目的第三控制；

允许向列表添加项目的第四控制；以及

允许显示项目的预览图像的第五控制。

25．如权利要求 22 所述的方法，其特征在于，还包括：

创建具有启用用户界面对话框和用户界面向导的功能的第三名字空间。

26．如权利要求 25 所述的方法，其特征在于，所述第三名字空间包括：

允许打开和保存文件和文件夹的第一对话框；

允许向光盘进行写入的第一向导；以及

方便通过电子邮件发送图像的第二向导。

27．如权利要求 22 所述的方法，其特征在于，所述第一名字空间包括：

允许向上下文菜单进行添加的第一功能；

允许标识应用定义的缩略图的第二功能；以及

允许当显示关于一个或多个文件或文件夹的信息时执行计算的第三功能。

28．如权利要求 22 所述的方法，其特征在于，所述第二名字空间包括：

允许在所述桌面上显示边条的第一功能；以及

允许在所述桌面上显示应用定义的通知的第二功能。

29．一种方法，其特征在于，包括：

调用一个或多个第一功能以使用用户界面的控制；以及

调用一个或多个第二功能以扩展所述用户界面的功能。

30．如权利要求29所述的方法，其特征在于，还包括：

调用一个或多个第三功能以扩展所述用户界面的桌面的功能。

31．如权利要求30所述的方法，其特征在于，扩展所述桌面的功能的功能包括：

允许在所述桌面上显示边条的一个或多个功能；以及

允许在所述桌面上显示应用定义的通知的一个或多个功能。

32．如权利要求29所述的方法，其特征在于，还包括：

调用一个或多个第三功能以使用所述用户界面的对话框和向导。

33．如权利要求32所述的方法，其特征在于，使用所述用户界面的对话框和向导的功能包括：

使用对话框以允许打开和保存文件和文件夹的一个或多个功能；

使用向导以允许向光盘进行写入的一个或多个功能；以及

使用另一向导以方便通过电子邮件发送图像的一个或多个功能。

34．如权利要求29所述的方法，其特征在于，使用所述用户界面的控制的功能包括：

封装存储用户经验的一个或多个功能；

允许以应用定义的方式显示项目的一个或多个功能；

允许向列表添加项目的一个或多个功能；以及

允许显示项目的预览图像的一个或多个功能。

35．如权利要求29所述的方法，其特征在于，扩展所述用户界面的功能的功能包括：

允许向上下文菜单进行添加的一个或多个功能；

允许标识应用定义的缩略图的一个或多个功能；以及

允许当显示关于一个或多个文件或文件夹的信息时执行计算的一个或多个功能。

36．一种方法，其特征在于，包括：

接收对使用用户界面的控制的一个或多个第一功能的一个或多个调用；以及

接收对扩展所述用户界面的桌面的功能的一个或多个第二功能的一个或多个调用。

37．如权利要求36所述的方法，其特征在于，还包括：

接收对扩展所述用户界面的功能的一个或多个第三功能的一个或多个调用。

38．如权利要求37所述的方法，其特征在于，扩展所述用户界面的功能的功能包括：

允许向上下文菜单作出添加的一个或多个功能；

允许标识应用定义的缩略图的一个或多个功能；

允许当显示关于一个或多个文件或文件夹的信息时执行计算的一个或多个功能。

39．如权利要求36所述的方法，其特征在于，还包括：

接收对使用所述用户界面的对话框和向导的一个或多个第三功能的一个或多个调用。

40．如权利要求39所述的方法，其特征在于，使用所述用户界面的对话框和向导的功能包括：

使用对话框允许打开和保存文件和文件夹的一个或多个功能；

使用向导允许向光盘进行写入的一个或多个功能；以及

使用另一向导方便通过电子邮件发送图像的一个或多个功能。

41．如权利要求36所述的方法，其特征在于，使用所述用户界面的控制的功能包括：

封装存储用户经验的一个或多个功能；

允许以应用定义的方式显示项目的一个或多个功能；

允许向所述桌面的边条添加项目的一个或多个功能；

允许向列表添加项目的一个或多个功能；以及

允许显示项目的预览图像的一个或多个功能。

42．如权利要求36所述的方法，其特征在于，扩展所述桌面的功能的功能包括：

允许在所述桌面上显示边条的一个或多个功能；以及

允许在所述桌面上显示应用定义的通知的一个或多个功能。

用于计算机平台的编程接口

技术领域

本发明涉及软件以及软件的开发。本发明尤其涉及方便应用程序和计算机硬件使用软件平台的编程接口。

背景技术

在很久以前，计算机软件被分类成“操作系统”软件或“应用”软件。广泛而言，应用是要为计算机用户执行具体任务的软件，如解数学方程或支持文字处理。操作系统是管理并控制计算机硬件的软件。操作系统的目标是使计算机资源对应用编程者可用，而同时隐藏实际控制硬件所必需的复杂性。

操作系统通过总称为应用程序接口或 API 的功能使得资源可用。也关于这些功能的单独一个使用术语 API。这些功能通常被按照它们向应用编程者提供什么资源或服务来分组。应用软件通过调用个别的 API 功能来请求资源。API 功能也起把操作系统所提供的消息和信息转送回应用软件的手段的作用。

除硬件中的变化之外，推动操作系统软件的发展的另一因素是简化并加快应用软件开发的期望。应用软件开发可以是一个令人感到畏缩的任务，它有时候需要几年的开发时间，使用上百万行代码来创建复杂的程序。对于诸如各种版本的 Microsoft Windows®操作系统等流行的操作系统来说，应用软件开发者每年书编写使用该操作系统的上千种不同的应用。需要一种相干且有用的操作系统基础来支持如此多的不同应用开发者。

通常可以通过使操作系统更复杂来将应用软件的开发变得更简单。即，如果一个功能对若干不同的应用程序有用，则一次性编写这一功能以包包含于操作系统中比需要许多软件开发者多次编写它以包含于许多不同的应用中更好。以此方式，如果操作系统支持许多应用所需要的较大范围的公共功能，则可以实现应用软件开发成本和时间的显著节省。

无论在操作系统和应用软件之间如何划界，很清楚，对于有用的操作系统，操作系统和计算机硬件以及应用软件之间的 API 与该操作系统本身的有效内部操

作一样重要。

在过去的几年中，因特网的全球使用以及一般性的网络技术已经改变了计算机软件开发者的前景。传统地，软件开发者集中在用于独立台式计算机或通过局域网（LAN）连接至有限数量的其它计算机的基于 LAN 的计算机的单点（single-site）软件应用。这类软件应用通常被称为“收缩包装（shrink wrapped）”产品，因为这些软件以收缩包装包的形式进行销售。应用使用较好定义的 API 来访问计算机的底层操作系统。

由于因特网的发展并获得了普遍的接受，该行业开始认识到在万维网（或简称 web）的不同位置上主控应用的能力。在网络化世界中，来自任何地方的客户机能够向主宿在不同位置的基于服务器的应用提交请求，并在零点几秒内接收回响应。然而，这些 web 应用通常使用最初为独立的计算机器或本地网络化的计算机开发的同一操作系统平台来开发。不幸的是，在一些情况下，这些应用不足以转移到分布式计算方式。完全未使用支持无限数量的相互连接的计算机的观念来构造底层平台。

为适应向由因特网引进的分布式计算环境的转移，微软公司开发了一种网络软件平台，称为“.NET”框架（读作“点 Net”）。Microsoft®.NET 是用于连接人、信息、系统和设备的软件。该平台允许开发者创建在因特网上执行的 web 服务。这一动态转移由微软的.NET™框架的一组 API 功能实现。

由于.NET™框架的使用变得越来越普遍，已确定了提高平台的效率和/或性能的方法。发明人开发了一组唯一的编程接口功能来允许这些提高的效率和/或性能。

发明内容

本发明描述了一种用于计算机平台的编程接口。

依照某些方面，该编程接口可包括以下几组服务的一个或多个：与可重复使用的用户界面控制有关的第一组服务、与用户界面对话框和用户界面向导有关的第二组服务、与扩展用户界面功能有关的第三组服务以及与扩展用户界面的桌面的功能有关的第四组服务。

附图说明

贯穿附图使用相同的标号来标识相同的特征。

图 1 示出了客户机使用常规协议通过因特网访问 web 服务的网络体系结构。

图 2 是用于网络平台的软件体系结构的框图，它包括了一种应用程序接口（API）。

图 3 是该 API 支持的唯一的名字空间以及各种 API 功能的功能类的框图。

图 4 是可执行软件体系结构的全部或部分的示例性计算机的框图。

图 5、6、7、8、9、10、11、12、13、14、15 和 16 示出了编程接口的各种示例实现。

具体实施方式

本揭示着眼于用于开发者可在其上构建 web 应用和服务的网络平台的编程接口，如应用程序接口（API）。更具体地，描述了用于使用诸如由微软公司创建的.NET™框架（.NET™ Framework）等网络平台的操作系统的示例性 API。.NET™框架是用于在分布式计算环境中实现的 web 服务和 web 应用的软件平台。它代表了下一代因特网计算，使用了开放通信标准以在协作执行具体任务的松散联系的 web 服务之间进行通信。

在所描述的实现中，网络平台使用 XML（可扩展标记语言），它是一种描述数据的开放标准。XML 由万维网联盟（W3C）管理。XML 用于定义 web 页和商业对商业文档中的数据元素。XML 使用一种与 HTML 类似的标签结构；然而，HTML 定义了如何显示元素，而 XML 定义了那些元素包含什么。HTML 使用预定义的标签，而 XML 允许由页面的开发者定义标签。由此，实际上可以标识任意的数据项，允许 web 页起数据库记录一样的功能。通过使用 XML 和其它开放协议，如简单对象访问协议（SOAP），网络平台允许集成适合用户需求的大范围的服务。尽管结合 XML 和其它开放标准描述了这里所述的本发明的实施例，这对所要求权利的本发明的操作并非必需。其它等效可行的技术足以实现此处所描述的本发明。

如这里所使用的，短语应用程序接口或 API 包括采用方法或功能调用的传统接口以及远程调用（如，代理、承接（stub）关系）和 SOAP/XML 调用。

示例性网络环境

图 1 示出了可以在其中实现诸如.NET™框架等网络平台的网络环境 100。网络环境 100 包括代表性 web 服务 102(1)、…、102(N)，它们提供了可通过网络 104

（如因特网）访问的服务。web 服务，总称为 102，是可重新使用且可在网络 104 上在程序上交互的可编程应用组件，一般通过行业标准 web 协议，如 XML、SOAP、WAP（无线应用协议）、HTTP（超文本传输协议）以及 SMTP（简单邮件传输协议）来实现，但是还可以使用通过网络与 web 服务进行交互的其它手段，如远程过程调用（RPC）或对象中介服务（broker）类型技术。web 服务可以是自描述的，并通常按照消息的格式和排序来定义。

web 服务 102 可由其它服务（如由通信链路 106 所表示）或软件应用，如 web 应用 110（如由通信链路 112 和 114 所表示）直接访问。示出每一 web 服务 102 包括执行软件来处理对具体服务的请求的一个或多个服务器。这类服务通常维护储存要向请求者提供的信息的数据库。web 服务可以被配置成执行各种不同的服务的任一种。web 服务的示例包括登录验证、通知、数据库存储、股票报价、位置目录、映射、音乐、电子钱包、日历/日程安排、电话清单、新闻和信息、游戏、购票等等。web 服务可以彼此组合，并与其它应用组合来构建智能交互式体验。

网络环境 100 还包括代表性客户机设备 120(1)、120(2)、120(3)、120(4)、…、120(M)，它们使用 web 服务 102（如由通信链路 122 所表示）和/或 web 应用 110（如由通信链路 124、126 和 128 所表示）。客户机也可使用标准协议相互通信，如由客户机 120(3)和 120(4)之间的示例性 XML 链接 130 所表示的那样。

客户机设备，总的标号为 120，可以以许多不同的方式来实现。可能的客户机实现的示例包括但不限于：便携式计算机、固定计算机、平板（tablet）PC、电视机/机顶盒、无线通信设备、个人数字助理、游戏控制台、打印机、复印机以及其它智能设备。

web 应用 110 是设计成在网络平台上运行且当处理并服务来自客户机 120 的请求时可使用 web 服务 102 的应用。web 应用 110 包括运行在编程框架 132 之上的一个或多个软件应用 130，它们在一个或多个服务器 134 或其它计算机系统上执行。注意，web 应用 110 的一部分可实际上驻留在一个或多个客户机 120 上。可选地，web 应用 110 可与客户机 120 上的其它软件协调以实际完成其任务。

编程框架 132 是支持由应用开发者开发的应用和服务的结构。它通过支持多语言来允许多语言开发和无缝集成。它支持开放协议，如 SOAP，并封装了底层操作系统和对象模型服务。该框架为多编程语言提供了一种健壮且安全的执行环境，并提供安全、集成的类库。

框架 132 是一种多层体系结构，它包括应用程序接口（API）层 142、公共语言运行时间（CLR）层 144 以及操作系统/服务层 146。这一分层体系结构允许对各层作出更新和修改而不影响该框架的其它部分。公共语言说明（CLS）140 允许各种语言的设计者编写能够访问底层库功能的代码。说明 140 的作用是语言设计者和库设计者之间的联系，它可以用来提升语言的互操作性。通过遵守 CLS，以一种语言编写的库可以被其它语言编写的代码模块直接访问，以实现以一种语言编写的代码模块和以另一种语言编写的代码模块之间的无缝集成。CLS 的一种示例性详细实现在由 ECMA TC39/TG3 的参与者创建的 ECMA 标准中有描述。读者可以通过 www.ecma.ch 访问 ECMA web 站点。

API 层 142 给出应用 130 可调用来访问由层 146 提供的资源和服务的功能组。通过展现用于网络平台的 API 功能，应用开发者能够为分布式计算系统创建充分利用网络资源和其它 web 服务的 web 应用，而不需要理解这些网络资源实际上如何操作或如何变为可用的复杂的相互作用。此外，web 应用可以以任意数量的编程语言来编写，并被翻译成由公共语言运行时间 144 所支持的中间语言，并作为公共语言说明 140 的一部分包括在内。这样，API 层 142 可为广泛且不同的各种应用提供方法。

此外，框架 132 可以被配置成支持由远离主控该框架的服务器 134 执行的远程应用所发出的 API 调用。代表性应用 148(1)和 148(2)分别驻留在客户机 120(3)和 120(M)上，它们可通过经网络 104 直接或间接向 API 层 142 作出调用来使用 API 功能。

也可以在客户机装置上实现该框架。客户机 120(3)表示框架 150 在客户机上实现的情况。该框架可与基于服务器的框架 132 相同，或为客户机目的而修改。可选地，在客户机是有限或专用功能设备，如蜂窝电话、个人数字助理、手持式计算机或其它通信/计算设备的情况下，可以精简基于客户机的框架。

开发者的编程框架

图 2 更详细地示出了编程框架 132。公共语言说明（CLS）层 140 支持以各种语言 130(1)、130(2)、130(3)、130(4)、…、130(K)编写的应用。这些应用语言包括 Visual Basic、C++、C#、COBOL、Jscript、Perl、Eiffel、Python 等等。公共语言说明 140 规定了特征的一个子集或关于特征的规则，如果遵循这些规则，则允许各种

语言进行通信。例如，某些语言不支持给定类型（如，“int*”类型），而该类型可由公共语言运行时间 144 支持。在这一情况下，公共语言说明 140 不包括该类型。另一方面，由所有或大多数语言支持的类型（如“int[]”类型）包括在公共语言说明 140 中，从而库开发者可以自由地使用这一类型，并且确保语言能够处理该类型。这一通信能力导致了以一种语言编写的代码模块和以另一种语言编写的代码模块之间的无缝集成。由于不同的语言特别能较好地适合于特定的任务，语言之间的无缝集成允许开发者利用将特定的代码模块用于以不同语言编写的代码模块的能力为该代码模块选择特定的语言。公共语言运行时间 144 允许具有跨语言继承性的多语言开发，并为多编程语言提供了一种健壮且安全的执行环境。关于公共语言说明 140 和公共语言运行时间 144 的更多信息，指示读者阅读 2000 年 6 月 21 日提交的题为“Method and System for Compiling Mutiple Languages）”（编译多语言的方法和系统）（序列号 09/598,105）以及 2000 年 7 月 10 日提交的“Unified Data Type System and Method”（统一数据类型系统和方法）（序列号 09/613,289）的共同待批的申请，这两个申请通过引用而被结合于此。

框架 132 封装了操作系统 146(1)（如，Windows®操作系统）和对象模型服务 146(2)（如组件对象模型（COM）或分布式 COM）。操作系统 146(1)提供了常规功能，如文件管理、通知、事件处理、用户界面（如，开窗口、菜单、对话框等）、安全、鉴别、验证、进程和线程、存储器管理等等。对象模型服务 146(2)提供了与其它对象的接口来执行各种任务。向 API 层 142 作出的调用被交付给公共语言运行时间层 144，用于由操作系统 146(1)和/或对象模型服务 146(2)本地执行。

API 142 将 API 功能分组成多个名字空间。名字空间本质上定义了类、接口、代表、枚举和结构的集合，总称为“类型”，它提供了一组具体的相关功能。类表示具有引用赋值语义的管理堆分配数据。代表是面向对象的功能指针。枚举是表示命名常数的一种特殊的值类型。结构表示具有值赋值语义的静态分配数据。接口定义了其它类型可执行的约定。

通过使用名字空间，设计者能够将一组类型组织成一个分层的名字空间。设计者能够从该组类型创建多个组，每一组包含逻辑地展现相关功能的至少一个类型。在示例性实现中，组织应用 142 来包括三个根名字空间。应当注意，尽管在图 2 中仅示出了三个根名字空间，但是在 API 142 中也可以包括另外的根名字空间。API 142 中示出的三个根名字空间是：用于演示子系统的第一名字空间 200（包括

用于用户接口外壳（shell）的名字空间 202）、用于 web 服务的第二名字空间 204 以及用于文件系统的第三名字空间 206。然后可以向每一组分配一个名字。例如，可以向演示子系统名字空间 200 中的类型分配名字“Windows”、向文件系统名字空间 206 中的类型分配名字“Storage”。可以在用于系统级 API 的单个“全局根”名字空间，如总体系统（System）名字空间下组织命名的组。通过选择顶层标识符并将其作为前缀，每一组中的类型可以由分层名字来容易地引用，该分层名字包括作为包含该类型的组的名字的前缀的所选择的顶层标识符。例如，文件系统名字空间 206 中的类型可以使用分层名字“System.Storage”来引用。以这一方式，个别的名字空间 200、204 和 206 变为从系统名字空间分支的主要部分，并可具有该个别名字空间在何处以如“System.”前缀之类的指示符作为前缀的指定。

演示子系统名字空间 200 属于编程和内容开发。它提供允许生成应用、文档、媒体演示和其它内容的类型。例如，演示子系统名字空间 200 提供一种允许开发者从操作系统 146(1)和/或对象模型服务 146(2)获取服务的编程模型。

外壳名字空间 202 属于用户接口功能。它提供了允许开发者在其应用中嵌入用户接口功能的类型，并还允许开发者扩充用户接口功能。

web 服务名字空间 204 属于用于启用诸如与在内联网上两个对等体之间操作的聊天应用一样简单的应用和/或与用于上百万个用户的可伸缩 web 服务一样复杂的应用之类的各种各样的 web 应用的创建的基础结构。所描述的基础结构是高度可变的，即只需要使用适合具体解决方案的复杂性的那些部分。该基础结构为构建不同规模和复杂性的基于消息的应用提供了基础。该基础结构或框架为基本消息通信、安全消息通信、可靠消息通信和已处理的（transacted）消息通信提供了 API。在下文所描述的实施例中，以仔细构造来平衡适用性、可用性、可扩充性和可定版本性的方式把关联的 API 被分解成名字空间层次。

文件系统名字空间 206 和存储有关。它提供了允许信息存储和检索的类型。

除框架 132 之外，提供了编程工具 220 来协助开发者构建 web 服务和/或应用。编程工具 220 的一个示例是 Visual Studio™，它是由微软公司提供的一套多语言编程工具。

根 API 名字空间

图 3 更详细地示出了演示子系统名字空间 200。在一个实施例中，依照分层命

名约定标识名字空间，在该约定中，名字串用句点来连接。例如，演示子系统名字空间 200 由根名字“System.Windows”来标识。在“System.Windows”名字空间内的是用于外壳（shell）的另一名字空间，它被标识为“System.Windows.Explorer”，进一步标识了用于控制的另一名字空间，称为“System.Windows.Explorer.Controls”。当了解了这一命名约定，以下提供了演示子系统名字空间 200 的综述，尽管可以使用其它命名约定来达到相等的效果。

外壳（Shell）名字空间 202（“System.Windows.Explorer”）包括支持用户界面功能的类和 API，以允许用户探索并导航到一组端点，如存储系统中的项目、FTP 位置或 DAV（分布式授权和版本）位置、控制面板中的控制面板小应用程序（applet）、应用等等。这种导航可涉及打开一个或多个文件夹，以及打开文件夹内的文件夹。外壳名字空间 202 允许开发者在其应用内嵌入这类功能，并进一步允许开发者扩展这种探索和导航功能。在各种版本的 Windows®操作系统中，这一用户界面通常被称为“Explorer”（资源管理器）。

在某些实施例中，外壳名字空间 202 的用户界面功能允许用户探索并导航的存储系统是用于组织、搜索和共享所有种类的信息的活动存储平台。该平台定义了一种丰富的数据模型，构建在关系存储引擎之上，支持灵活的编程模型，并提供了一组用于监控、管理并操纵数据的数据服务。数据可以是基于文件或非文件数据，并且数据通常被称为“项目”。文件系统扩展了通常由文件系统提供的功能，因为它也处理非文件数据的项目，如个人联系人、事件日期和电子邮件消息。这类文件系统的一个示例是“WinFS”文件系统。

外壳名字空间 202 的用户界面功能允许用户探索并导航的存储系统的通用数据存储实现了一种支持驻留在存储（store）中的数据的组织、搜索、共享、同步和安全的数据模型。该数据存储中的存储信息的基础单元被称为项目。该数据模型提供了一种用于声明项目和项目扩展的机制，用于建立项目之间的关系，并用于组织文件夹和类别中的项目。

项目是可储存的信息的单元，与简单的文件不同，它是具有通常由存储系统展现给用户或应用程序的所有对象所支持的一组基本特性的对象。项目也具有通常由所有项目类型所支持的特性和关系，包括允许引入新特性和关系的特征。这一特特性和关系数据也可以被称为与项目关联的元数据。如后文更详细讨论的那样，元数据可依照项目修饰模式来储存。这一项目修饰模式可指示向用户呈现项目的适当

方式。

项目是用于常用操作，如复制、删除、移动、打开、打印、备份、恢复、重复等的对象。项目是可以被储存并检索的单元，并且由存储平台操纵的可储存的信息的所有形式作为项目、项目的特性或项目之间的关系而存在，它们中的每一个将在下文详细讨论。项目表示真实世界和易理解的数据单元，如联系人、人、服务、位置、（所有各种排序的）文档，等等。

项目是独立的对象；从而，如果删除了项目，该项目的所有特性也被删除。类似地，当检索项目时，所接收到的是该项目以及包含在该项目的元数据内的其所有特性。某些实施例可以使得在检索具体项目时请求特性子集；然而，许多这样的实施例默认当检索时以其直接和继承的特性提供该项目。此外，也可以通过向该项目类型的现存特性添加新特性来扩展该项目的特性。这些“扩展”此后为该项目的真实特性，并且该项目类型的子类型可自动包括扩展的特性。扩展也可以被称为与文件关联的元数据。

项目组可被组织成特殊的项目，称为项目文件夹（item Folder）（不要与文件夹混淆）。然而，与大多数文件系统不同，一个项目可属于一个以上的项目文件夹，使得当在一个项目文件夹中访问并修订一个项目时，然后可以从另一项目文件夹直接访问这一修订的项目。本质上，尽管对一个项目的访问可以从不同的项目文件夹进行，但是实际所访问的事实上是完全相同的项目。然而，项目文件夹不必要拥有所有其成员项目，或可简单地结合其它文件夹共同拥有项目，使得一个项目文件夹的删除不必导致项目的删除。

项目组也可以被组织成类别（Category）。类别在概念上与项目文件夹不同，项目文件夹可包括不相互关联（即，没有公共描述的特征）的项目，而类别中的每一项目具有为该类别描述的公共类型、特性或值（“公共性”），并且该公共性形成了该类别中其它项目之间或与其它项目的关系。此外，具体文件夹中的项目的成员资格不必要基于该项目的任一具体方面，对于某些实施例，具有与类别绝对相关的公共性的所有项目在硬件/软件接口系统级上可自动变为该类别成员。在概念上，类别也可以被认为是虚拟项目文件夹，其成员资格基于具体查询的结果（如在数据库的上下文中），并且满足该查询的条件的项目（由该类别的公共性定义）将包括该类别的成员资格。

与文件、文件夹和目录相反，本发明的项目、项目文件夹和类别本质上不是

特性上“物理的”（physical），因为它们不具有物理载体的等效概念，并且因此项目可在一个以上这样的位置上存在。项目在一个以上项目文件夹位置中存在的能力以及被组织成类别的能力超越现有技术提供了硬件/软件接口级上的增强且强化的数据操纵以及存储结构能力。

项目也可包含关系信息，允许确定两个或多个项目之间的关系。关系是一种二元关系，其中，一个项目被指定为源，另一项目为目标。源项目和目标项目通过关系相关联。

通用数据存储中的项目由外壳浏览器呈现给用户，外壳浏览器提供了一种用户界面（使用外壳名字空间 202 的类和 API），允许用户查看硬件/软件接口并与之交互。通用数据存储中的每一项目依照通用数据模式来储存。该模式包括用于描述项目的机制，称为类型关联。每一类型关联具有外壳中的基本表示；通过依照类型关联储存项目，外壳能够至少依照基本或默认显示视图来显示项目。

类型关联是与项目关联的特性；当在通用数据存储中放置数据时，必须声明与该数据关联的一个或多个特性，以确定该项目是何类型。这些特性可以作为与该数据关联的元数据包括在内。外壳具有一组默认类型关联，它表示了必须为项目声明的最基本和最小的特性。

除特性声明之外，与项目相关联的元数据可包括指示外壳应如何修饰项目的呈现的数据。在本情况下，修饰可被认为是如何向用户表示该项目的“提示”。元数据可以依照项目修饰模式来储存。项目修饰模式定义了外壳可用来呈现该项目的项目修饰视图。例如，项目修饰数据可描述项目的最重要声明特性。这些“高值”特性可以是用于在外壳中呈现所最期望的。

为呈现项目，项目修饰数据可指示适用于呈现该项目的一组视图区（view field）。视图区是声明的特性的投影，且公共视图区可包括“标题”、“作者”、“创建日期”或“最后编辑”。外壳包括一组标准视图区，且独立软件销售商（ISV）可定义适合呈现其数据的视图区。当开发新项目类型时，ISV 可以将他们定义的项目特性映射到外壳的视图区，或者可提供其自己的视图区。

例如，某一项目数据可包含歌曲数据。声明的特性组可包括如歌曲标题、艺术家、录制日期、专辑、歌曲长度和适合这类歌曲项目的其它声明之类的特性。项目修饰数据可指示当在外壳中呈现该项目时应当向用户显示视图区“标题(Title)”、“艺术家（Artist）”和“专辑（Album）”。

项目修饰可以是外壳所支持的显示的任一方面。一些常见的其它项目修饰是，例如，数据格式化、默认排序次序和默认图标大小。另外，项目修饰数据可描述在显示给定项目中所使用的常见控制。例如，分级（Rating）区可使用以一列星号来表示分级的分级控制。项目修饰数据可描述适合用于项目的任务或动词。术语“任务”和“动词”描述了关于项目所采取的某一动作，并且这些术语可以互换使用。例如，“编辑”或“预览”可以是与项目关联的适当的任务/动词。外壳还可以被配置成在用户选择执行关于项目的动作时装入支持这些任务的应用。

项目修饰视图足够完全呈现给定项目或同类的项目组，包括具有类似的项目修饰视图的项目。为使用不同的项目修饰模式显示项目，外壳提供了依照外壳修饰视图呈现项目的外壳视图模式。外壳视图模式允许外壳或 ISV 为给定的异类数据组声明适当的视图。

所选择的用于在外壳修饰视图中表示的项目可包括公共特性。外壳修饰视图可接受各种各样的公共特性。例如，外壳视图模式可定义用于显示公共和合适区域的“图片”视图，以及用于所有已知图片类型（如.GIF、.JPEG、.BMP、.TIFF 等）的元数据。外壳视图模式不考虑给定项目修饰的冲突显示属性，并依照外壳视图模式呈现每一图片项目。作为另一示例，外壳可提供在适当的列周围优化的“文档”外壳视图以及用于由典型生产性应用生成的，如文字处理文档、电子表格或数据库之类的项目的元数据，即使这些项目的每一个的项目修饰可彼此大不相同。当安装后来的文档类型时，外壳视图能够依照一致的外壳视图呈现这些新项目，即使在首先创建该视图时未考虑该新类型也是如此。

外壳视图模式可提供各种各样的显示属性，并且 ISV 可希望提供这些外壳视图。显示属性可包括而不限于：预览窗格的大小、要在预览窗格中显示的元数据、要使用的自定义控制、以及适合所呈现的项目的任务和动词。

外壳可依照探索器显示视图来呈现项目。“探索器”（explorer）可以被称为一种存储应用，并可由外壳或 ISV 提供。例如，探索器可令用户能够查看、查询、导航、进入任务或组织数据存储中所选择的项目。术语“探索器”不应当意味着所显示的项目驻留在何处，可以与“探索器”互换地使用诸如“活动中心”、“查看器”和“库”等术语来描述存储应用。

示例性的探索器模式层次包括作为项目视图模式的底层。项目视图模式提供了表示项目所需要的基本显示，当需要时，探索器视图模式可在其显示元素上延迟

或后退。

探索器视图模式还包括外壳视图模式和探索器修饰。探索器修饰将探索器作为一个整体修饰，并提供诸如相异色彩和标记（branding）元素的显示元素。这些探索器修饰在探索器所提供的各种视图中保持。各种各样的显示属性可适合探索器修饰。例如，与探索器项目关联的数据查询或任务/动词可适用于使用探索器的显示。显示的任务常与能够执行该任务的应用相耦合。

探索器视图模式可任选地包括一个外壳视图模式或多个外壳视图模式。外壳视图模式可以被配置成为探索器项目的子集提供外壳视图。例如，探索器可被配置成向用户显示歌曲项目。可包括第一外壳视图模式来提供专辑的显示，可包括第二外壳视图来提供歌曲音轨的显示。以这一方式，两种类型的项目将在探索器内具有合适的视图。如上所述，使用外壳视图涉及对可任选地共享公共特性的一组项目的呈现。

探索器也可依赖于包括在外壳内的外壳视图。如果所选择的用于在探索器内呈现的项目不被该探索器所包括的任一外壳视图支持，则该外壳可提供在该探索器内使用的合适的外壳视图。类似地且如上所述，该探索器也可退回到由外壳所提供的项目显示视图或默认显示视图。这一功能确保可由外壳显示的任一项目也能够在探索器内显示。探索器能够被配置成顺从由显示模式提供的这些外壳或可依赖于它们，以例如为未预期的数据提供显示。

外壳名字空间 202 支持用户界面功能，它允许以一致的方式操纵并交互项目而不管储存该项目的基础数据（如文件内容、图像数据、联系人信息数据等）的位置如何。从应用设计者的观点来看，由外壳名字空间 202 支持的功能至少部分地得到较好的开发经验，因为可以以一致的方式来对待这些项目，尽管基础数据储存在不同的位置。当使用外壳名字空间 202 时，向设计者提供了相干经验，不论这些不同数据存储位置的可能存在如何。

外壳名字空间 202 定义了另外的名字空间，包括控制（Controls）名字空间 302、对话框（Dialogs）名字空间 304、内插式附件（Addin）名字空间 306 和桌面（Desktop）名字空间 308。外壳名字空间中的这些另外的名字空间 302、304、306 和 308 的每一个包括提供各种功能的一个或多个服务或组件，如控制、对话框和/或处理器，如后文所更详细讨论的那样。

外壳名字空间 202 定义了可由另外的名字空间 302、304、306 和 308 的每一

个使用的对象模型。这一对象模型包括以下对象：

• 探索器项目（ExplorerItem）对象（也称为项目（Item）对象），用于描述可经外壳向用户呈现的项目，如文件夹、栈、个别文件、个别 WinFS 项目（如专辑、歌曲、图片等）等等。

• 库（Library）对象，用于对照探索器项目描述查询。

• 视图区（ViewField）对象（也称为特性（Properties）对象），用于投影（如由探索器项目对象表示的）项目数据以向用户显示。存储系统中的特性和视图区对象之间定义的投影可基于探索器项目执行另外的计算或处理。例如，如果正在显示表示存储设备的文件夹且设计者希望在该文件夹内包括该存储设备上的大量空闲空间，则投影基于该存储设备的总容量和储存在该存储设备上的文件（被表示为探索器项目）的大小来计算空闲空间量。

• 存储收藏（StorageFavorite）对象，用于描述到动态列表的链接，它是库对象的一种持久形式。存储收藏对象是查询和向用户呈现该查询的结果的视图的组合。

对象模型更详细地在 2003 年 10 月 23 日递交的题为“System and method for presenting items to a user with a contextual presentation”（使用上下文呈现向用户呈现项目的系统和方法）的美国专利申请号（未定），代理人档案号 MFCP.109834（MS# 306923.01）中讨论，该申请通过引用而被结合于此。

控制（Controls）名字空间 302（“System.Windows.Explorer.Controls”）定义了可由应用使用的可重复使用的用户界面控制的集合。由于多个不同的应用可使用这些控制，这些控制可被视为可重复使用。可重复使用的控制的集合可简化应用的设计，因为应用设计者可依赖于这些控制而不需要设计他们自己的版本的控制。另外，通过使用这种可重复使用的控制的集合，可获得跨多个应用的一致外观和感受。

由控制名字空间 302 定义的控制允许操纵和/或显示用户界面的外壳，包括操纵和/或显示用户界面中的项目。如上所述，可在用户界面的外壳内显示的这些项目表示各种各样的数据的任一个，并且不仅仅是储存在文件系统中的传统“文件”。此外，由控制名字空间 302 定义的控制允许应用设计者在用来自各种各样的位置的数据进行工作时使用相同的控制，从而得到一致且相干的设计经验。

控制名字空间 302 定义了以下可重复使用的控制：

• 探索器视图（ExplorerView）控制用于封装存储用户经验（storage user

experience）。存储用户经验被定义成跨项目使用关系、查询等工作。例如，这一存储用户经验允许终端用户基于特性值发出查询、堆栈具有公共特性值的 WinFS 项目、允许从非堆栈项目拖曳到堆栈（如通过将特性断言中的特性值设为栈中所存在的值）等等。探索器视图控制也可用于描述用于显示的具有可以在先前的操作系统（也称为传统系统），如 Microsoft® Windows®操作系统（如 Windows®操作系统的 Windows® XP 或 Windows® 98 版本）中找到的相同的外观的文件夹。

• 探索器项目视图（ExplorerItemView）控制用于显示探索器项目。探索器项目的布局和设计可由应用设计者定义。操作系统也可定义在可由应用设计者定义的布局和设计上施加约束的用户经验，使得由应用设计者定义的布局和设计与由操作系统定义的用户经验一致。从而，探索器项目视图控制允许设计者确定如何向用户显示探索器项目的数据（如以列、行或某一其它配置）以及向用户显示什么数据。

• 篮控制（BasketControl）控制用于允许向边条（sidebar）添加项目（边条在后文另外讨论）。可以通过使用光标控制设备选择一个项目并将其拖至边条以“拖放”的方式向边条添加项目，在边条上可以放下该项目（如通过释放光标控制按钮）。篮控制控制在 2003 年 10 月 13 日递交的题为“Extensible Creation And Editing Of Integrated Collections”（综合集的可扩展创建和编辑）的美国专利申请号（未定），代理人档案号 003797.00695（MS# 304631.1）中有更详细的描述，该申请通过引用而被结合于此。

• 列表制作（ListMaker）控制用于允许向列表和组添加项目。可以通过使用光标控制设备选择一个项目并将其拖至合适的列表或组上以“拖放”的方式创建项目的列表或组，在列表或组上可以放下项目（如，通过释放光标控制按钮）。例如，项目可以是歌曲，列表可以是播放列表，项目可以是图像，列表可以是相册，等等。

• 预览图像控制（PreviewImageControl）组件用于允许向用户呈现项目的预览图像。当选择一个项目时，可以向用户呈现预览图像，预览图像通常大于该图像的缩略图而小于用户界面的整个显示区域。

对话框（Dialog）名字空间 304（“System.Windows.Explorer.Dialogs”）定义了可由应用使用的用户界面对话框和向导的集合。类似于控制名字空间 302 中的控制，对话框名字空间 304 中的对话框和向导可以被重复使用，使得多个不同的应用可以使用这些对话框和向导。这一可重复使用的对话框和向导的集合帮助跨多个应用维持一致的外观和感受。

由对话框名字空间 304 定义的对话框和向导允许在用户界面的外壳内执行动作。如上所述，这些外壳可以基于可以表示各种各样数据的任一个的项目，并不仅仅是储存在文件系统中的传统“文件”。此外，由对话框名字空间 304 定义的对话框和向导允许应用设计者在用来自各种各样位置的数据工作时使用相同的对话框和向导，得到一致且相干的设计经验。

对话框名字空间 304 定义了以下对话框和向导：

• 打开/保存（Open/Save）对话框用于允许打开和保存文件和文件夹。这些对话框也是可扩展的，允许应用设计者扩展对话框的功能。例如，可以扩展这些对话框来使用关于应用设计者期望使用的文件和/文件夹的各种另外的元数据。

• CD/DVD 烧录（CD/DVD Burn）向导用于允许向光盘，如 CD 和/或 DVD 进行写入。硬件设计者可以提供允许向 CD 和/或 DVD 进行写入的其自己的向导，包括其自己的硬件专用步骤。

• 发送照片（Send Photos）向导用于协助用户通过电子邮件发送图像。该向导允许用户将图像调整为特定的大小。该向导可将图像调整为被认为是适合电子邮件发送的特定的大小，或者可选地允许用户从两个或多个不同的大小中选择。

内插式附件（Addin）名字空间 306（“System.Windows.Explorer.Addin”）定义了用于扩展用户界面功能的基类和接口的集合。内插式附件名字空间 306 允许应用设计者添加扩展用户接口的功能的可控制代码。例如，这一扩展允许应用设计者在他们认为合适时自定义用户界面的部分。

内插式附件名字空间 306 允许扩展用于显示项目的用户界面的外壳，它表示了各种各样的数据且不仅仅是储存在文件系统中的传统“文件”。此外，内插式附件名字空间 306 允许应用设计者以相干的方式扩展用户界面的外壳，提供了更为可控的且改进的设计经验。

各种功能的可控制代码由内插式附件名字空间 306 支持，如：

• 上下文菜单处理器。例如，可以通过在显示的项目上右键点击来访问上下文菜单。上下文菜单处理器允许设计者定义其自己的要添加到这些上下文菜单的条目。

• 缩略图提取器。缩略图提取器允许应用设计者（如为显示的文件、文件夹和/或其它项目）开发其自己的缩略图，并向操作系统标识这些缩略图以向用户显示。

·用于组成/分解从 WinFS 到探索器的投影的处理器。这些处理器允许应用设计者在向用户显示关于文件和/或文件夹的信息时执行各种计算（如允许计算存储设备上的空闲空间）。

桌面（Desktop）名字空间 308（“System.Windows.Explorer.Desktop”）定义了允许应用设计者扩展向用户显示的桌面的功能的功能集合。桌面名字空间可以是外壳名字空间 202 的一部分（如“System.Windows.Explorer.Desktop”），或可选地可以是演示子系统名字空间 200 的一部分而不是外壳名字空间 202 的一部分（如“System.Windows.Desktop”）。桌面名字空间包括边条和通知中涉及的元素。在桌面名字空间中包括边条和通知元素向用户提供了边条和通知的集中式控制。例如，可以由用户一次性设置关于来自所有应用的通知的规则，而不是需要用户通过访问使用通知的每一应用的规则定义来导航。

一般而言，边条在集成交互式外围知晓显示（integrated inactive peripheral awareness display）内提供了动态通信访问和信息知晓，其中，动态地跟踪并接收规定的通信联系人和信息元素并在进行的基础上向用户提供。在某些实施例中，通过在沿常规显示设备的一边的持久显示条中一个或多个列中显示的至少一个可自定义的动态缩略图来提供这一能力。此外，在另外的实施例中，在显示屏的任一一个或多个部分，包括整个显示屏上显示缩略图。在其中覆盖了整个显示屏的实施例在边条用于具有相对小显示区域的设备上时尤其有用，这些设备例如手持式或掌上计算设备、蜂窝电话或具有有限显示区域的任一其它电子设备。

例如，每一可自定义的动态缩略图表示了具体的通信联系人（如，具体的个人、商家、组织或其它实体）或用户感兴趣的具体的信息元素。这类信息元素包括，如，共享文件或文件夹何时被修改、共享数据库或工作空间中的信息何时改变、电子邮件状态、日历、因特网 web 页、天气条件、约会、日程安排、统计信息、股票报价、交通信息或用户感兴趣的任一其它因特网或网络可访问信息。

上述动态缩略图一般包括描述感兴趣的联系人或信息的“平铺块”（tile）（也称为“标签”（ticket））和用于显示由该平铺块表示的任何通信联系人或信息的专用“查看器”的组合。例如，每一平铺块和查看器可以是存储系统中可经外壳名字空间 202 提供的用户界面功能访问的项目。使用一个或多个“服务”来自动交互、跟踪或接收由每一平铺块描述的信息的当前状态和/或通信联系人的状态。然后通过驻留在在用于图形和/或文字地显示项目的交互式外围知晓接口的“载体”中主

控每一平铺块来动态地提供信息的当前状态和通信联系人的状态。外围知晓接口以减少对用户的潜在分心和中断的方式显示信息和/或通信联系人。

一般而言，由诸如 XML 数据文件等数据结构表示平铺块。每一平铺块包括要由该平铺块表示哪些信息和通信联系人的指示，以及到表示用于访问和/或交互信息或通信联系人的多种常规手段的任一种的具体服务的指针。这些服务自动或手动地从预定义或用户可定义的服务库中选择。特别地，不同的服务表示提供用于访问、接收、检索和/或交互任一常规信息、信息源或通信联系人的功能的共享代码或功能。这些服务在它们单独或组合使用的意义上共享，并可以由一个或多个平铺块同时使用。因此，应当注意，在某些实施例中，组合地使用了多个服务来提供与任何常规信息、信息源或通信联系人的交互。

服务的一个示例是通过连接至常规 MAPI 服务器监控电子邮件文件夹所必需的功能。服务的另一示例是发送或接收电子邮件消息的功能。相关的服务提供了用于通过任一多种常规方法，如即时消息通信或对等通信模式，与联系人进行通信或传输信息的功能。服务的另一示例是将文本文件从一种语言转化到另一种语言的功能。服务的再一示例是监控数据库所必需的功能。服务的又一示例包括用于从 web 站点或远程服务器接收或检索数据的功能。很清楚，用于与任何信息、信息源或通信联系人进行交互的任一方法可被实现为由一个或多个平铺块使用的共享服务。

此外，如上所述，每一平铺块的指令包括到具有显示由该平铺块表示的任何类型的信息或通信联系人的能力的多种专用查看器之一的指针。换言之，每一平铺块表示了用户期望连同用户期望如何查看特定信息或联系人的定义一起跟踪的信息或联系人的组合，以及使用任一多种服务来访问信息或联系人且与其交互的能力。这种访问或交互可以通过任一常规通信协议本地或跨本地内联网、外联网、有线或无线网络、因特网等实现。

查看器图形地将平铺块显示为具有通过一个或多个服务检索的信息或联系人的可调整大小的缩略图或图表大小的窗口。具体地，查看器能够动态地显示具有文本、音频或图形信息的平铺块，包括静止或活动图像，或文本、音频或图形信息的任一组合。例如，一种查看器类型能够显示联系人信息，即“个人平铺块”，另一种能够显示具体的电子邮件信息，如接收的消息数或来自特定源的消息数，另一查看器被设计成与数据库进行交互，以缩略图提供来自该数据库的具体信息的概述。查看器类型的更多示例包括能够显示静止图像、视频图像、通信状态概述、数据库

查询结果等的查看器。很清楚，任一类型的查看器可以被设计成与对应类型的信息关联，以确保可以显示任一可能的信息。

边条（sidebar）在题为“A system and process for providing dynamic communication access and information awareness”（用于提供动态通信访问和信息知晓的系统和过程）的美国专利申请号 10/063,296 中更详细地描述，该申请通过引用而被结合于此。

名字空间 308 的边条功能允许应用设计者向边条添加平铺块。

通知（Notification）（如可听的或可视的通知）是用户上下文环境系统的一部分，依照某些实施例，它包括被比较来决定如何处理通知的三个元素。第一个元素是用户的上下文环境（可由操作系统和已扩展它的任意程序来提供）。第二个元素是用户的规则和偏好。第三个元素是通知本身（包含诸如可匹配用户的规则的数据和特性等元素）。

用户上下文环境系统由操作系统和声明用户的上下文环境的其它程序操作，其后系统代理用户的上下文环境和规则。通知由调入系统的其它程序引发。比较用户的上下文环境、规则和通知的元素，然后确定应当对通知作出什么动作。对通知作出什么动作的各种选项的示例包括拒绝(如，如果不允许该通知绘制或发出噪声，并且该通知从未被用户所见）、推迟（如，保留通知直到用户上下文环境改变或用户规则规定它随后适合传送）、传送（如，依照用户上下文环境和规则允许传送通知）和路由（如，用户规则指示该通知应当被切换到另一系统，而不管该通知是否被允许在本系统中传送）。

一般而言，用户可以在被认为是“不可用”的状态，在这一情况下，通知既不被传送也不被保留，直到用户变为“可用”为止。例如，如果用户正在运行全屏应用，该用户可以被认为是不可用的。或者，用户可以是“可用”，但是是在需要修改通知以适合该用户的状态下。例如，如果用户正在听音乐或在会议中，用户可指示通知应当被传送到用户屏幕，但是它所发出的声音应当更安静或不发出任何声音。

如上所述，用户上下文环境可部分地确定是否在用户屏幕上显示通知。当显示通知时，可以基于用户上下文环境内的某些梯度来显示。换言之，可以规定绘制的通知的形式的侵入的不同级别。例如，常规通知能够自由地弹出到客户区域并暂时遮盖窗口。如果用户在轻微限制的上下文环境中，通知可以自由显示，但是仅以

不入侵的方式，例如不允许它在另一窗口上绘制。作为另一示例，在用户正在运行最大化应用的一个实施例中，默认设定可以是这意味着该环境是轻微限制的，并且用户明确地作出了他们希望该应用获取整个客户区域的状态。在该设定中，通知仍被允许绘制，但是仅令其出现在边条内。换言之，通知绘制形式中的这一类型的降低的侵入性减少了通知的影响，并总体上减少了感知负荷。

用户上下文环境系统在 2003 年 3 月 26 日提交的题为“System and Method Utilizing Test Notifications”（使用测试通知的系统和方法）的美国专利申请号 10/402,179 中有更详细的描述，该申请通过引用而被结合于此。

名字空间 308 的通知功能允许应用设计者创建其自己的通知，并当用户可能感兴趣的事情发生时（如接收到电子邮件、接收到即时消息通信等）在桌面上显示它们来通知用户。名字空间 308 的通知功能还为用户提供了中央位置来定义他们对来自多个应用的通知的规则和偏好，使用户不用多次设定相同的规则和偏好（对多个应用的每一个设定一次）。

另外，联系人（Contacts）名字空间定义了联系人的控制和对话框的集合。这种联系人信息可以作为项目储存在存储系统中。“联系人”一般指任一人、组、组织、商家、住户或其它类型的可标识实体。“联系人信息”一般指对应于联系人且可以被认为与标识、联系、访问联系人、与联系人交流或通信相关的任一信息。联系人信息由应用使用来执行期望的功能，如，发送电子邮件、启动电话呼叫、访问 web 站点、启动游戏会话、执行金融交易等等。联系人信息的非限制示例包括名字、别名、电话号码、电子邮件地址、家庭地址、即时消息通信（IM）地址和 web 地址。联系人信息也可指诸如联系人状态等其它类型的信息。例如，指示联系人当前在线或在电话线路上的信息也可以被概括地认为是联系人信息。

联系人名字空间可以是演示子系统名字空间 200 的一部分，但不是外壳名字空间 202 的一部分，如“System.Windows.Contacts”、“System.Windows.Controls”（图 3 的控制 310）或“System.Windows.Collaboration”。可选地，联系人名字空间可以是外壳名字空间 202 的一部分。联系人名字空间扩展了 System.Windows 名字空间的功能，但是不需要被实现为外壳的一部分（即，不需要为 System.Windows.Explorer 名字空间 302 的一部分）。

联系人名字空间允许联系人信息被用户输入一次，而仍可由多个不同的应用访问。由此，用户没有多次输入相同的联系人信息（对多个应用的每一个输入一次）

的负担。

联系人名字空间中的控制和对话框包括：

• 联系人挑选工具（Contact Picker）对话框，用于允许用户从其储存的联系人中选择或挑选联系人。例如，允许用户选择要向其发送电子邮件消息的联系人，或进行即时消息通信的联系人。例如，用户可以在作为项目储存在存储系统中的所有的用户联系人的列表中进行浏览，并选择那些联系人的一个或多个。联系人挑选工具对话框在 2002 年 12 月 19 日提交的题为“Contact Picker”（联系人挑选工具）的美国专利申请号 10/324,746 中有更详细的描述，该申请通过引用而被结合于此。

• 联系人文本框（Contact Textbox）控制，用于允许用户以自由的形式输入诸如名字等信息（如通过键入名字），并且当输入联系人信息时，该控制将该信息解析成来自用户储存的联系人的一个或多个联系人。例如，可以在下拉菜单中显示解析出的信息，或可以自动将该信息输入到用户以自由的形式输入名字的输入行中。

• 联系人（Contact）控制，用于向用户显示联系人信息。该控制不支持联系人信息的编辑或自由形式条目。

示例性名字空间成员

本部分包括了描述可由示例性名字空间（如图 3 的演示子系统名字空间 200）展现的成员的示例的多个表格。这些展现的名字空间可包括，例如，类、接口、枚举和代表（delegate）。可以理解，这些示例中所描述的成员仅为示例，可选地可由名字空间展现其它成员。

System.Windows.Explorer.Controls

下标列出了由 System.Windows.Explorer.Controls 名字空间展现的成员的示例。

类

DefaultCommandEventArgs	包含 DefaultCommandEvent（默认命令事件）事件的事件数据。
ExplorerView	允许用户查看关于文件夹的内容的信息。
FolderSelectionChangedEventArgs	包含 FolderSelectionChangedEvent（文件夹选择改变事件）事件的事件数据。

IncludeItemEventArgs	包含 IncludeItemEvent（包括项目事件）事件的事件数据。
ItemVerbModifyEventArgs	初始化 ItemVerbModfiyEventArgs（项目动词修改事件自变量）类的新实例。
NavigationCompleteEventArgs	包含 NavigationCompleteEvent（导航完成事件）事件的事件数据。
NavigationFailedEventArgs	包含 NavigationFailedEvent（导航失败事件）事件的事件数据。
NavigationPendingEventArgs	包含 NavigationPendingEvent（导航待决事件）事件的事件数据。

枚举

FolderViewFlags	指示 ExplorerView（探索器视图）的视图特性。
FolderViewMode	指示 ExplorerView（探索器视图）的视图模式。

代表（delegates）

DefaultCommandEventHandler	表示处理 DefaultCommandEvent 的方法。
FolderSelectionChangedEventHandler	表示处理 FolderSelectionChangedEvent 的方法。
IncludeItemEventHandler	表示处理 IncludeItemEventde 的方法。
ItemVerbModifyEventHandler	表示处理 ItemVerbModifyEvent 的方法。
NavigationCompleteEventHandler	表示处理 NavigationCompleteEvent 的方法。
NavigationFailedEventHandler	表示处理 NavigationFailedEvent 的方法。
NavigationPendingEventHandler	表示处理 NavigationPendingEvent 的方法。

System.Windows.Explorer.Dialogs

下表列出了由 System.Windows.Explorer.Dialogs 名字空间展现的成员的示例。

类	
ColorDialog	
CommonDialog	
DialogControlItemSelectedEventArgs	
FileDialog	用作 OpenFileDialog（打开文件对话框）和 SaveFileDialogBase（保存文件对话框基础）的父类的抽象类。
FileDialogCheckButton	表示可放置在 FileDialog（文件对话框）上的复选框控制。
FileDialogComboBox	
FileDialogContainerControlBase	用作 FileDialogComboBox（文件对话框组合框）、FileDialogOpenDropDown（文件对话框打开下拉）、FileDialogRadioButtonGroup（文件对话框单选按钮组）和 FileDialogToolbarMenu（文件对话框工具条菜单）的父类的抽象类。
FileDialogControlBase	用作 FileDialogCheckButton（文件对话框复选按钮）、FileDialogEditBox（文件对话框编辑框）、FileDialogPushButton（文件对话框按钮）和 FileDialogContainerControlBase（文件对话框载体控制基础）的父类的抽象类。
FileDialogControlItem	
FileDialogEditBox	表示可以放置在 FileDialog 上的文本框控制。
FileDialogOpenDropDown	
FileDialogPushButton	表示可以放置在 FileDialog 上的按钮控制。
FileDialogRadioButtonGroup	
FileDialogToolbarMenu	
FileOkEventArgs	

FileType	
OpenFileDialog	允许用户选择一个或多个文件打开。该类不能被继承。
SaveAsFileDialog	使用户能够选择保存文件的位置并指定文件名。该类不能被继承。
SaveFileDialog	允许用户选择保存文件的位置。该类不能被继承。
SaveFileDialogBase	用作 SaveAsFileDialog（保存为文件对话框）和 SaveFileDialog（保存文件对话框）的父类的抽象类。

枚举

FileDialogLayout

TileAttributes

代表（delegates）

DialogControlItemSelectedEventHandler

FileOkEventHandler

System.Windows.Desktop

下表列出了由 System.Windows.Desktop 名字空间展现的成员的示例。该名字空间包含边条和通知中涉及的元素。

类

AnalogClockPanel

Appbar

AreaButton

BaseComTile

BaseSidebarClockSetting

BaseTile	用作自定义边条平铺块实现的父类的抽象类。

BasketControl	
CalendarElement	
CalendarImages	
ChildrenWontFitArgs	
ClockHacks	
ClockPanel	
DataSourceEventArgs	表示从数据源传递的事件数据。
DigitalDateTimeElement	
DragButton	
DragControlWindow	
DraggableButton	
DragWindow	
DropArgs	
DropEventArgs	
ExtraSpaceArgs	
FillAlphaPresenter	
FillImageResourcePresenter	
FillPanel	
FillPresenter	
Flyout	
FlyoutLinkClickEventArgs	表示响应于起源于边条平铺块的飞出视图的底部找到的可任选链接的事件传递到RMAActionEventHandler（RMA动作事件处理器）的事件数据。
FlyoutPresenter	
FlyoutStuff	
FocusableButton	
FocusWithinWorkaroundHelper	
FolderContentsChangedEventArgs	表示响应于文件夹内容中的变化传递到FolderContentsChangedEventHandler（文件夹内

	容改变事件处理器）的事件数据。
GlobalSetting	
HackBorder	
HackImage	
ImageButton	
ImageResource	
ImageResourcePresenter	
ItemControl	
ItemToolbarControl	
ListMakerControl	
MenuStuff	
NativeResource	
NativeResourceHelper	
NativeResources	
NativeResourceTypeConverter	
NormalButton	
Notification	表示系统在用户界面（UI）的片断中发送给用户的消息及其关联的数据。
NotificationArea	
NotificationButton	定义出现在通知中的按钮。
NotificationClickedEventArgs	包含与通知窗口中的点击事件关联的数据。
NotificationContext	声明应用当前是否在特定的状态。该状态被用作用户上下文环境的定义的一部分。
NotificationTimerPresenter	
NotifyCompleteEventArgs	
NotifyWindow	
OptionsButton	
OuterTile	
OverflowableCollection	
OverflowableControlCollection	

OverflowPanel	
OverflowPresenter	
OverFlowWrapper	
PanelInfo	
ProgressBar	
QuickLaunchTile	
RMAActionEventArgs	表示由多功能最小化应用（Rich Minimized Application）（RMA）生成的事件数据。由 RMAActionEventHandler（RMA 动作事件处理器）使用。
RMAData	
SetDataArrayEventArgs	表示由数据数组生成并发送到 SetDataArrayEventHandler（设置数据数组事件处理器）的事件数据。
Sidebar	
SidebarAlarmClock	
SideBarClock	
SidebarClockSettings	
SlideShowTile	
StackAlphaPresenter	
StackImageResourcePresenter	
StartButton	
SyncHelper	
SyncItemBar	
SyncTile	
Taskbar	
TaskChevron	
TaskGroup	
TaskItem	
TaskList	

TaskOverflow

TaskOverflowableControlCollection

TaskPresenter

TestTile

TestTimeZone

TextElementFontInfo

Theme

ThemeHelper

ThemeHelperOld

ThemeImage

ThemeImageTypeConverter

ThemeResources

Tile

TileCollection

FileOverflow

TileThumb

TileThumbDotNet

Timer

TimerStuff

TimeZone

ViewStatusControl

WebHostEventArgs

WindowOrigin

WindowOriginOld

WindowStuff

接口

ICOMDataSourceHandler 用于创建以自由代码实现的数据源并与其进行通信。

IOverflow

IOverflowList

ISidebar	提供对主控了个别平铺块的边条的访问。主边条负责打开并关闭飞出、显示快捷方式菜单并响应于涉及个别平铺块的事件。
ISidebarAlarm	

枚举

AlarmState	
BasketFlags	
DragWindowPos	
InvokeFolderActionEnum	当用户调用文件夹的 ItemControl（项目控制）时决定要作出的动作的枚举。用于 InvokeFolderAction（调用文件夹动作）和 ItemControlInvokeFolderAction（项目控制调用文件夹动作）。
RMAAction	用于 Action（动作）特性来定义事件动作的常量。
SelectionModeEnum	

代表（delegates）

ChildrenWontFitHandler	
DocumentCompleteEventHandler	
DropEventHandler	
DropHandler	
DummyDelegateToSetupAppDomainProperly	
ExtraSpaceHandler	
FlyoutClosingEventHandler	
FlyoutLinkClickEventHandler	表示处理 FlyoutLinkClickEvent（飞出链接点击事件）事件的方法。
FolderContentsChangedEventHandler	
NavigateErrorEventHandler	
NotificationClickedEventHandler	表示处理 NotificationClickedEvent（通知点击事件）事件的方法。

NotifyCompleteEventHandler	表示处理 NotifyCompleteEvent（通知完成事件）事件的方法。
RMAActionEventHandler	表示处理 RMAActionEvent（RMA 动作事件）事件的方法。
SetDataArrayEventHandler	表示处理 SetDataArrayEvent（设置数据数组事件）事件的方法。

System.Windows.Controls

提供了由外壳组件使用的类和接口。

下表列出了由 System.Windows.Controls 名字空间展现的成员的示例。

类

ContactControl

ContactPickerDialog

ContactPropertyRequest

ContactPropertyRequestCollection

ContactSelection

ContactSelectionCollection

ContactTextBox

ContactTextBoxSelectionChangedEventArgs

ContactTextBoxTextChangedEventArgs

ContactTextBoxTextResolvedEventArgs

IncludeContactEventArgs

IteratedEventArgs

Iterator

OKCancelApplyEventArgs

OKCancelApplyStrip

枚举

ContactControlPropertyPosition

ContactPickerDialogLayout

ContactPropertyCategory

ContactPropertyType

ContactType

OKCancelApplyType

代表（delegations）

ContactTextBoxSelectionChangedEventHandler

ContactTextBoxTextChangedEventHandler

ContactTextBoxTextResolvedEventHandler

IncludeContactEventHandler

IteratedEventHandler

OKCancelApplyEventHandler

OpenedEventHandler

示例性计算系统和环境

图 4 示出了在其中实现编程框架 132（全部或部分）的适用的计算环境 400 的一个示例。计算环境 400 可在这里描述的计算机和网络体系结构中使用。

示例性计算环境 400 仅为计算环境的一个示例，并非建议对计算机和网络体系结构的使用或功能的范围的局限。也不应将计算环境 400 解释为对示例性计算环境 400 中示出的任一组件或其组合具有依赖或需求。

框架 132 可以使用众多其它通用或专用计算系统环境或配置来实现。适合使用的众所周知的计算系统、环境和/或配置包括但不限于，个人计算机、服务器计算机、多处理器系统、基于微处理器的系统、网络 PC、小型机、大型机、包括任一上述系统或设备的分布式计算环境等等。该框架的简化或子集版本也可以在诸如如蜂窝电话、个人数字助理、手持式计算机或其它通信/计算设备之类的有限资源的客户机中实现。

框架 132 可以在计算机可执行指令的一般上下文环境中描述，计算机可执行指令如程序模块，由一个或多个计算机或其它设备执行。一般而言，程序模块包括

例程、程序、对象、组件、数据结构等等，它们执行特定的任务或实现特定的抽象数据类型。框架 132 也可以在分布式计算环境中实践，其中，任务由通过通信网络连接的远程处理设备来执行。在分布式计算环境中，程序模块可以位于本地和远程计算机存储媒质中，如存储器存储设备。

计算环境 400 包括计算机 402 形式的通用计算设备。计算机 402 的组件可包括但不限于，一个或多个处理器或处理单元 404、系统存储器 406 以及将包括处理器 404 的各类系统组件耦合至系统存储器 406 的系统总线 408。

系统总线 408 代表若干种总线结构类型的一个或多个，包括存储器总线或存储器控制器、外围总线、加速图形端口以及使用各类总线结构的处理器或局部总线。作为示例而非局限，这类结构包括工业标准体系结构（ISA）总线、微通道体系结构（MCA）总线、增强型 ISA（EISA）总线、视频电子标准协会（VESA）局部总线以及外围部件互连（PCI）总线，也称为夹层（Mezzanine）总线。

计算机 402 通常包括各种计算机可读媒质。这类媒质可以是可由计算机 402 访问的任一可用媒质，包括易失性和非易失性媒质、可移动和不可移动媒质。

系统存储器 406 包括以易失性存储器形式的计算机可读媒质，如只读存储器（ROM）410，和/或非易失性存储器形式的计算机存储媒质，如随机存取存储器（RAM）412。基本输入/输出系统（BIOS）414 包括如在启动时帮助在计算机 402 内的元件之间传输信息的基本例程，储存在 ROM 412 中。RAM 410 通常包含处理单元 404 立即可访问和/或当前正在操作的数据和/或程序模块。

计算机 402 也可包括其它可移动/不可移动、易失性/非易失性计算机存储媒质。作为示例，图 4 示出了对不可移动、非易失性磁性媒质（未示出）进行读写的硬盘驱动器 416、对可移动、非易失性磁盘 420（如“软盘”）进行读写的磁盘驱动器 418 以及对可移动、非易失性光盘 424，如 CD ROM、DVD-ROM 或其它光媒质进行读写的光盘驱动器 422。硬盘驱动器 416、磁盘驱动器 418 和光盘驱动器 422 的每一个通过一个或多个数据媒质接口 423 连接到系统总线 408。可选地，硬盘驱动器 416、磁盘驱动器 418 和光盘驱动器 422 可以通过一个或多个接口（未示出）连接到系统总线 408。

盘驱动器及其关联的计算机可读媒质为计算机 402 提供了计算机可读指令、数据结构、程序模块和其它数据的非易失性存储。尽管本示例示出了硬盘 416、可移动磁盘 420 和可移动光盘 424，可以理解，也可以使用可储存可由计算机访问的

数据的其它类型的计算机可读媒质，如磁带盒或其它磁存储设备、闪存卡、CD-ROM、数字多功能盘（DVD）或其它光存储器、随机存取存储器（RAM）、只读存储器（ROM）、电可擦除可编程只读存储器（EEPROM）等，来实现示例性计算系统和环境。

多个程序模块可储存在硬盘416、磁盘420、光盘424、ROM 412和/或RAM 410中，作为示例包括，操作系统426、一个或多个应用程序428、其它程序模块430和程序数据432。操作系统426、一个或多个应用程序428、其它程序模块430和程序数据432的每一个（或其某一组合）可包括编程框架132的元素。

用户可以通过诸如键盘434和指点设备436（如“鼠标”）等输入设备向计算机402输入命令和信息。其它输入设备438（未具体示出）可包括话筒、操纵杆、游戏垫、圆盘式卫星天线、串行端口、扫描仪和/或其类似物。这些和其它输入设备通过耦合至系统总线408的输入/输出接口440连接到处理单元404，但是也可以通过其它接口和总线结构连接，如并行端口、游戏端口或通用串行总线（USB）。

监视器442或另一类型的显示设备也通过接口，如视频适配器444连接到系统总线408。除监视器442之外，其它输出外围设备可包括诸如扬声器（未示出）和打印机446等组件，通过输入/输出接口440连接到计算机402。

计算机402可以在使用到一个或多个远程计算机，如远程计算设备448的逻辑连接的网络化环境中操作。远程计算设备448可以是个人计算机、便携式计算机、服务器、路由器、网络计算机、对等设备或其它公共网络节点等。示出远程计算设备448为包括这里所描述的与计算机402有关的许多或所有元素和特征的便携式计算机。

计算机402和远程计算机448之间的逻辑连接被描述为局域网（LAN）450和一般广域网（WAN）452。这类网络环境常见于办公室、企业范围计算机网络、内联网以及因特网。

当在LAN网络环境中使用时，计算机402通过网络接口或适配器454连接至局域网450。当在WAN网络环境中使用时，计算机402通常包括调制解调器456或其它装置，用于通过广域网452建立通信。调制解调器456可以对计算机402是内置或外置的，通过输入/输出接口440或其它合适的机制连接至系统总线408。可以理解，示出的网络连接是示例性的，也可以使用在计算机402和448之间建立通信链路的其它手段。

在网络化环境中，如使用计算环境 400 所示出的，描述的与计算机 402 或其部分相关的程序模块可储存在远程存储器存储设备中。作为示例，远程应用程序 458 驻留在远程计算机 448 的存储器设备中。为说明目的，本发明示出应用程序和诸如操作系统等其它可执行程序组件为离散块，尽管可以认识到，这类程序和组件在不同的时刻驻留在计算设备 402 的不同存储组件中，并由计算机的数据处理器执行。

框架 132 和/或 150 的一种实现，尤其是框架 132 和/或 150 中包括的 API 或对框架 132 和/或 150 中包括的 API 作出的调用，可以储存在某一形式的计算机可读媒质中或在其之间传输。计算机可读媒质可以是可由计算机访问的任一可用媒质。作为示例而非局限，计算机可读媒质包括“计算机存储媒质”和“通信媒质”。“计算机存储媒质”包括以用于储存信息的任一方法或技术实现的易失性和非易失性，可移动和不可移动媒质，信息如计算机可读指令、数据结构、程序模块或其它数据。计算机存储媒质包括但不限于，RAM、ROM、EEPROM、闪存或其它存储器技术、CD-ROM、数字多功能盘（DVD）或其它光存储器、磁盒、磁带、磁盘存储或其它磁存储设备、或可以用来储存所期望的信息并可由计算机访问的任一其它媒质。

“通信媒质”通常在诸如载波或其它传输机制的已调制数据信号中包含计算机可读指令、数据结构、程序模块或其它数据。通信媒质也包括任一信息传送媒质。术语“已调制数据信号”指以对信号中的信息进行编码的方式设置或改变其一个或多个特征的信号。作为示例而非局限，通信媒质包括有线媒质，如有线网络或直接线缆连接，以及无线媒质，如声音、RF、红外线和其它无线媒质。上述任一的组合也应当包括在计算机可读媒质的范围之内。

可选地，框架的部分可以以硬件或硬件、软件和/或固件的组合来实现。例如，可以设计或编程一个或多个专用继承电路（ASIC）或可编程逻辑器件（PLD）来实现该框架的一个或多个部分。

可编程接口（或更简单地称为接口）可以被视为用于使代码的一个或多个片断与由代码的一个或多个其它片断提供的功能进行通信或访问的任一机制、过程、协议。可选地，编程接口可以被视为能够通信上耦合至其它组件的一个或多个机制、方法、功能调用、模块等的系统的组件的一个或多个机制、方法、功能调用、模块、对象等。上述语句中的术语“代码片断”包括代码的一个或多个指令或行，并包括，如，代码模块、对象、子例程、功能等等，无论应用的术语是什么、或代码片断是

否被单独编译、或代码片断是否被提供为源、中间物或对象代码、代码片断是否在运行时间系统或过程中使用、或它们是否位于同一或不同机器上或跨多个机器分布、或由代码片断表示的功能是否完全由软件、完全由硬件或硬件和软件的组合来实现。

概念上，编程接口可以被一般地查看，如图 5 或图 6 所示。图 5 示出了接口“接口 1”为管道，第一和第二代码片断通过该管道进行通信。图 6 示出了接口包括接口对象 I1 和 I2（可以是或不是第一和第二代码片断的部分），它们使系统的第一和第二代码片断通过媒质 M 进行通信。在图 6 中，可以认为接口对象 I1 和 I2 为同一系统的单独接口，并且也可以认为对象 I1 和 I2 加上媒质 M 构成了接口。尽管图 5 和 6 示出了双向流程以及该流程的每一侧上的接口，但是某些实现可仅具有一个方向上的信息流（或如下所述没有信息流），或仅在一侧具有接口对象。作为示例而非局限，诸如应用程序编程或程序接口（API）、进入点、方法、功能、子例程、远程过程调用和组件对象模型（COM）接口等术语包含在编程接口的定义之内。

这类编程接口的方面可包括第一代码片断向第二代码片断发送信息的方法（其中，“信息”以其最广泛的意义使用，并包括数据、命令、请求等等）；第二代码片断接收信息的方法；以及该信息的结构、序列、语法、组织、模式、定时和内容。在这一点上，只要信息以接口所定义的方式传输，底层传输媒质本身可以对接口的操作不重要，无论该媒质是有线还是无线，或两者的组合。在某些情况下，在常规意义上，当一个代码片断仅访问由第二代码片断执行的功能时，信息可不在一个或两个方向上传输，因为信息传输可以是或者通过另一机制（如，信息被放置在与代码片断之间的信息流分离的缓存、文件等中）或者不存在。这些方面的任一个或全部可能在给定的情况下是重要的，如，取决于代码片断是否是松散耦合或紧密耦合的配置中的系统的一部分，因此这里所列的应当被认为是说明性的而非限制。

编程接口的这一概念对本领域的技术人员来说是已知的，并且可以阅读上述本发明的详细描述而清楚这一概念。然而，有其它方法来实现编程接口，并且除非明显地排除，这些方法也包含在本发明的范围之内。这些其它方法可能看似比图 5 和 6 的示意图更精密或复杂，但是它们仍执行类似的功能来完成相同的整体结果。现在简要描述编程接口的某些说明性的替换实现方式。

A．分解

可以通过将通信分成多个离散的通信来间接地实现从一个代码片断到另一个代码片段的通信。这在图 7 和 8 中示意性地描述。如图所示，可以按照可分的功能组来描述某些接口。由此，可以分解图 5 和 6 的接口功能来达到相同的结果，如同可以在数学上提供 24，或 2 乘 2 乘 3 乘 2 一样。因此，如图 7 所示，可以细分由接口“接口 1”提供的功能以将该接口的通信变换成多个接口“接口 1A”、“接口 1B”、“接口 1C”等，而达到相同的结果。如图 8 所示，接口 I1 提供的功能可被细分成多个接口 I1a、I1b、I1c 等，而达到相同的结果。类似地，从第一代码片断接收信息的第二代码片断的接口 I2 可以被分解成多个接口 I2a、I2b、I2c 等。当分解时，包括在第一代码片断中的接口的数量不需要匹配包括在第二代码片断中的接口的数量。在图 7 或 8 的任一情况下，接口“接口 1”和 I1 的功能要旨分别与图 5 和 6 的保持相同。接口的分解也可遵从联合、通信和其它数学特性，使得分解较难识别。例如，操作的排序可以是不重要的，并且因此由接口执行的功能可以在达到该接口之前由另一段代码或接口较好地执行，或者由系统的单独组件执行。此外，编程领域的普通技术人员可以理解有各种方式来作出不同的功能调用而达到相同的结果。

B．重定义

在某些情况下，可能忽略、添加或重定义编程接口的某些方面（如参数），而仍达到预期的结果。这在图 9 和 10 中示出。例如，假定图 5 的接口“接口 1”包括功能调用 *Square(input, precision, output)*（平方功能），调用包括三个参数，*input*（输入）、*precision*（精度）和 *output*（输出），并且由第一代码片断向第二代码片断发出。如果中间的参数 *precision* 在给定的情形下无关紧要，如图 9 所示，它可以仅被忽略或甚至由 *meaningless*（无意义）（在这一情况下）参数来替换。也可以添加无关紧要的参数 *additional*（另外）。在任一情况下，只要输入被第二代码片断平方之后返回输出，就可以实现平方的功能。*precision* 对某一下行流或计算系统的其它部分来说可能是非常有意义的参数；然而，一旦认识到 *precision* 对计算平方的狭窄目的来说是不必需的，它就可以被替换或忽略。例如，不是传递一个有效的 *pricision* 值，而是在非不利地影响结果的情况下传递诸如出生日期等无意

义的值。类似地，如图 10 所示，接口 I1 由接口 I1'替换，它被重新定义来忽略或向接口添加参数。接口 I2 可类似地被重定义为接口 I2'，它被重定义来忽略不必要的参数，或可在别处被处理的参数。此处的要点是在某些情况下，编程接口可包括诸如参数等方面，它们对某一目来说不必要，因此可以忽略或重定义它们，或在别处处理它们以用于其它目的。

C．联机编码（inline coding）

合并两个单独的代码模块的一些或全部功能也是可行的，使得它们之间的“接口”改变形式。例如，图 5 和 6 的功能可以被分别转化到图 11 和 12 的功能。在图 11 中，图 5 的先前的第一和第二代码片断被合并成包含两者的一个模块。在这一情况下，这两个代码片断仍可以彼此通信，但是接口可以适用于更适合于单个模块的形式。由此，例如，正式的调用（Call）和返回（Return）语句（statement）将不再必需，但是依照接口“接口 1”的类似的处理或响应仍是有效的。类似地，如图 12 所示，图 6 的部分（或所有）接口 I2 可以联机地写入接口 I1 来形成接口 I1"。如所示，接口 I2 被划分成 I2a 和 I2b，并且接口部分 I2a 与接口 I1 一起联机编码来形成接口 I1"。对于具体的示例，考虑图 6 的接口 1 执行功能调用 *square(input, output)*，它由接口 I2 接收，在由第二代码片断处理由 *input* 传递的值（对其求平方）之后，它被使用 *output* 传递回已求平方的结果。在这一情况下，由第二代码片断执行的处理（对 *input* 求平方）可以由第一代码片断在不调用该接口的情况下执行。

D．脱离

可以通过将通信分成多个离散的通信来间接地完成从一个代码片断到另一个的通信。这在图 13 和 14 中示意性地描述。如图 13 所示，提供了中间件的一个或多个片断（脱离接口（Divorce Interface），因为它们从原始的接口脱离出功能和/或接口功能），以转化第一接口“接口 1”上的通信，使得它们符合不同的接口，在本情况下为“接口 2A”、“接口 2B”和“接口 2C”。这可以在这样一种情况中完成，例如，应用的已安装基础设计成依照“接口 1”协议与如操作系统进行通信，但是然后改变该操作系统来使用不同的接口，在本情况下为接口“接口 2A”、“接口 2B”和“接口 2C”。要点是改变了由第二代码片断使用的原始接口，使得它不在与第一代码片断所使用的接口兼容，因此使用中间物来使得旧接口和新接口

兼容。类似地，如图 14 所示，可以使用脱离接口 DI1 引入第三代码片断以从接口 I1 接收通信，并使用脱离接口 DI2 引入第三代码片断以向例如接口 I2a 和 I2b 发送接口功能，重新设计接口 I2a 和 I2b 以与 DI2 一起工作，但是提供相同的功能性结果。类似地，DI1 和 DI2 可共同工作以将图 6 的接口 I1 和 I2 的功能翻译成一新操作系统，而提供相同或类似的功能性结果。

E. 重写

再一种可能的变化是动态地重写代码，使用别的东西来替换接口的功能，而仍达到相同的总体结果。例如，可以有一种系统，其中，向执行环境（如由.Net 框架提供的环境、Java 运行时间环境或其它类似的运行时间类型环境）中的及时（Just-in-Time）（JIT）编译器或解释器提供以中间语言（Microsoft IL、Java ByteCode 等）呈现的代码片断。可以编写 JIT 编译器以动态地将通信从第一代码片断转化到第二代码片断，即，使它们符合第二代码片断（原始或不同的第二代码片断）可能需要的不同接口。这在图 15 和 16 中有描述。如图 15 中所看见的，这一方式类似于上述的脱离情形。它可以在这样一种情况下完成，例如，设计应用的已安装基础依照“接口 1”协议与操作系统进行通信，然后改变该操作系统以使用不同的接口。JIT 编译器可以用于使已安装基础应用的空中通信符合操作系统的新接口。如图 16 所描述的，可以应用这一动态重写接口的方法以进行动态分解，或者改变接口。

应当注意，上述通过替代实施例实现与接口相同或相似的结果的情形也可以以各种方式串行、并行或与其它中间代码组合。由此，上文给出的替代实施例并非相互排斥，而是可以被混合、匹配和组合以产生与图 5 和 6 中所呈现的一般情形相同或等效的情形。也应当注意，如同大多数编程构造，这里可能未描述达到与接口相同或相似的功能的其它类似的方式，但是它们仍由本发明的精神和范围来表示，即，应当注意，它至少部分地是由作为接口的值的基础的接口表示的功能或由其启用的有利结果。

总结

尽管以对结构特征和/或方法动作特定的语言描述了本发明，应当理解，所定义的本发明不必局限在所描述的具体特征或动作上。相反，描述具体特征和动作作

为实现本发明的示例性形式。

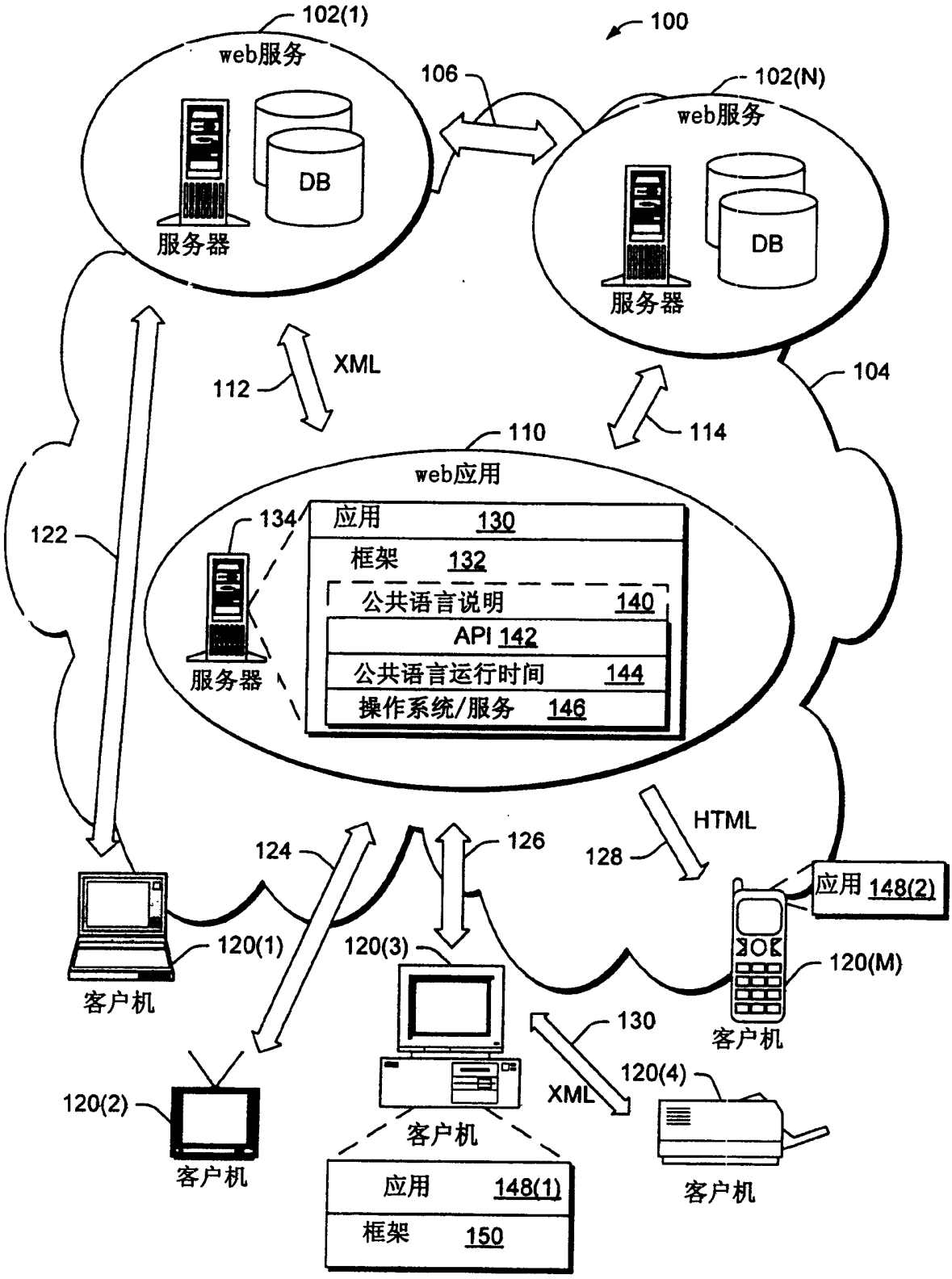

102(1)
web服务
100
106
102(N)
web服务
DB
服务器
DB
服务器
XML
112
104
114
110
web应用
122
134
应用 130
框架 132
公共语言说明 140
API 142
公共语言运行时间 144
操作系统/服务 146
服务器
HTML
128
124
126
应用 148(2)
120(1)
120(3)
120(M)
客户机
客户机
130
120(4)
XML
120(2)
客户机
客户机
客户机
应用 148(1)
框架 150

图 1

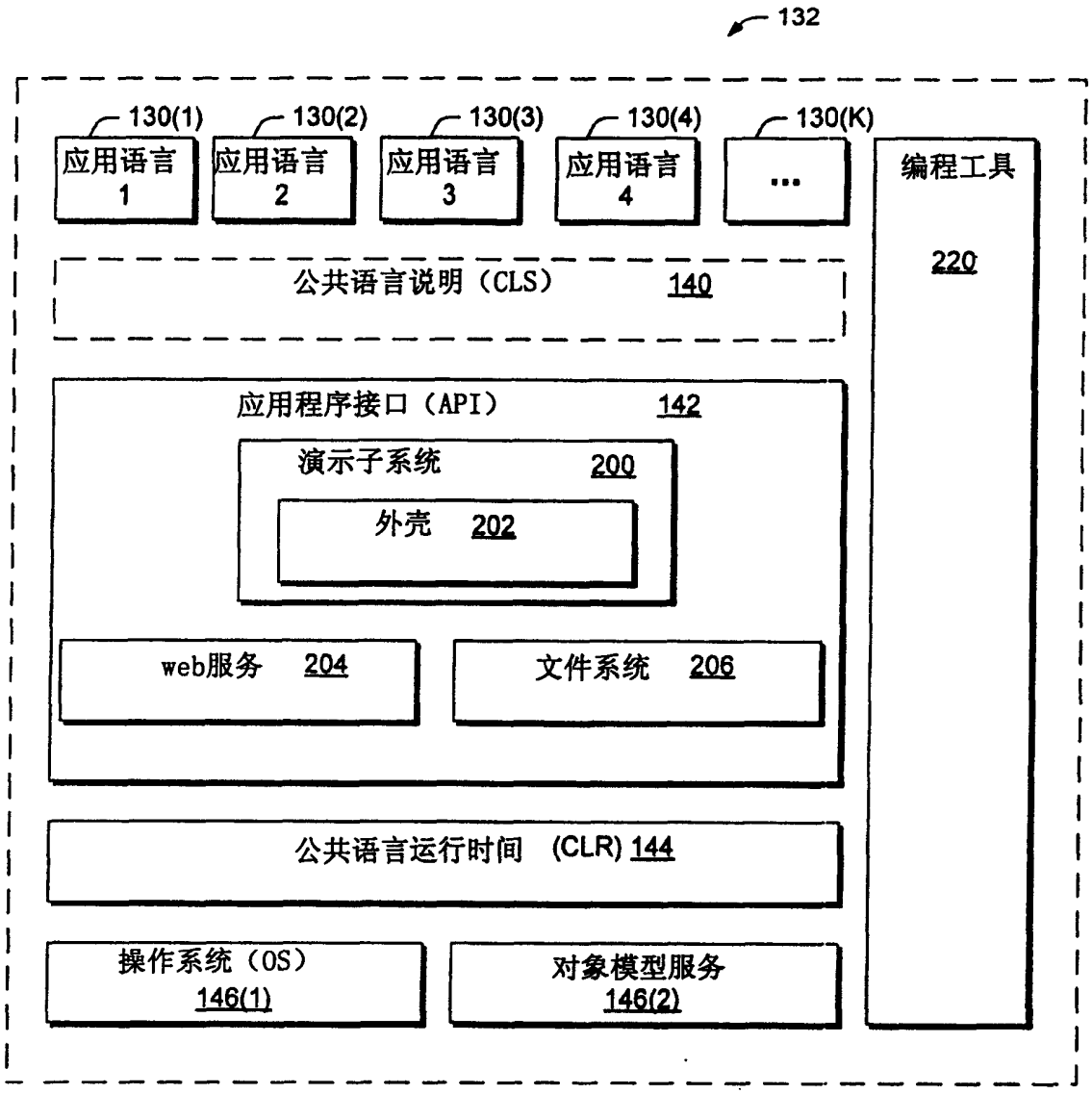

132
130(1)
130(2)
130(3)
130(4)
130(K)
应用语言 1
应用语言 2
应用语言 3
应用语言 4
...
编程工具
220
公共语言说明（CLS） 140
应用程序接口（API） 142
演示子系统 200
外壳 202
web服务 204
文件系统 206
公共语言运行时间 (CLR) 144
操作系统（OS） 146(1)
对象模型服务 146(2)

图 2

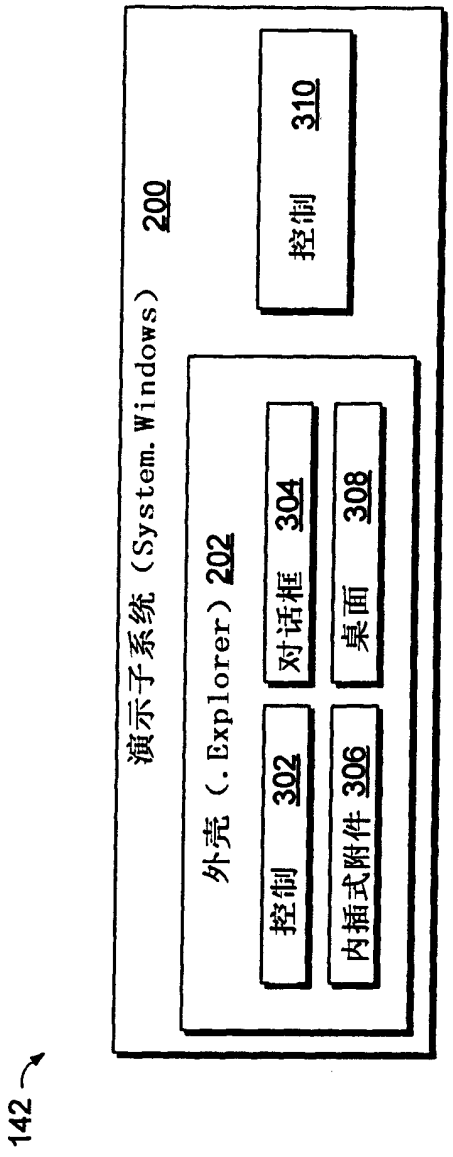

142
演示子系统（System.Windows） 200
外壳（.Explorer）202
控制 302
对话框 304
内插式附件 306
桌面 308
控制 310

图 3

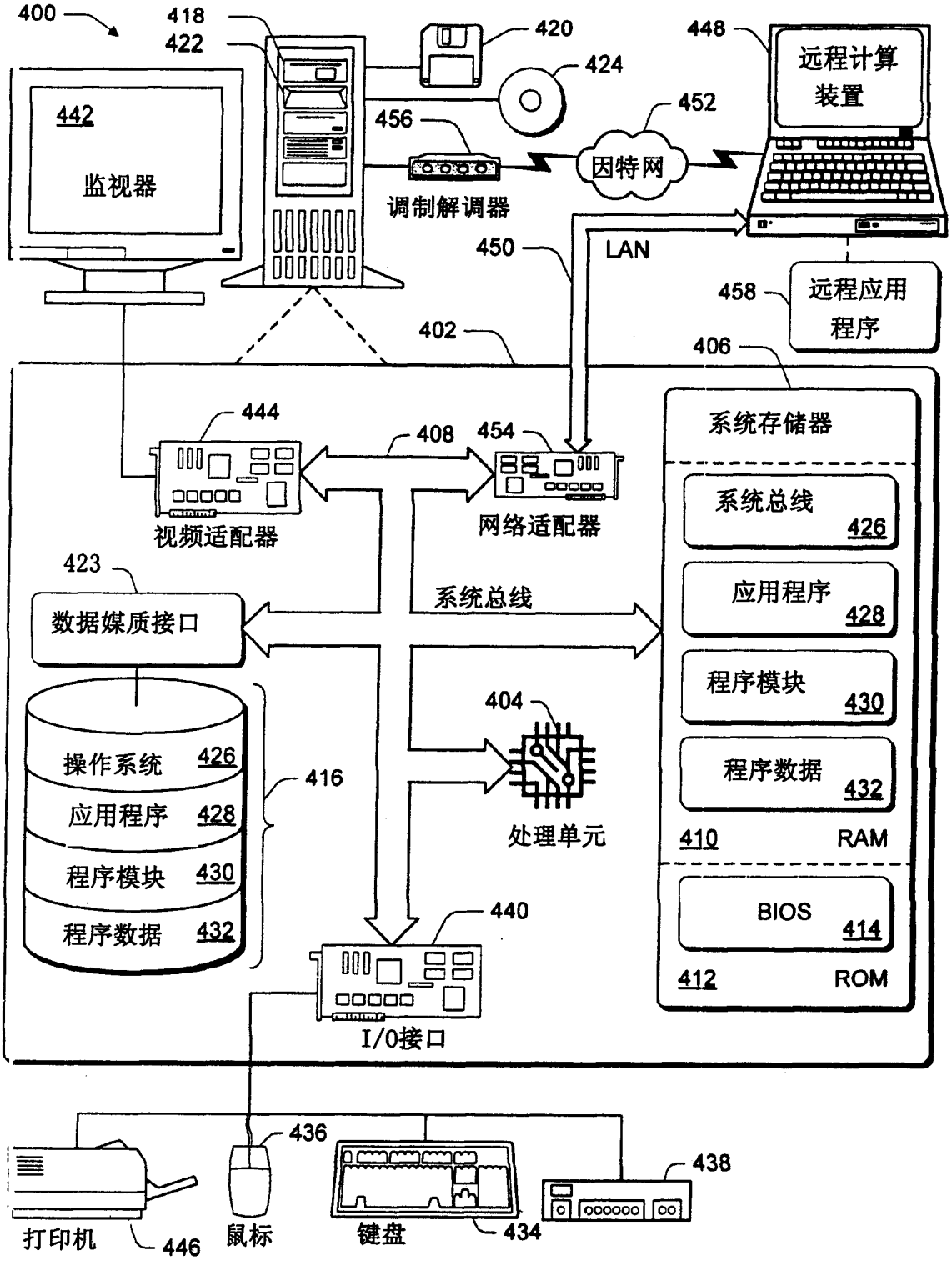

400
418
422
420
424
448
远程计算装置
442
监视器
456
452
因特网
调制解调器
450
LAN
458
远程应用程序
402
406
444
408
454
系统存储器
视频适配器
网络适配器
系统总线
426
423
应用程序
428
数据媒质接口
系统总线
程序模块
430
404
操作系统 426
应用程序 428
程序模块 430
程序数据 432
416
处理单元
程序数据
432
410
RAM
BIOS
414
440
412
ROM
I/O接口
436
438
打印机
446
鼠标
键盘
434

图 4

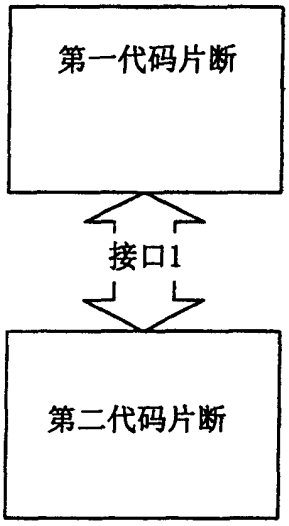

第一代码片断
接口1
第二代码片断

图 5

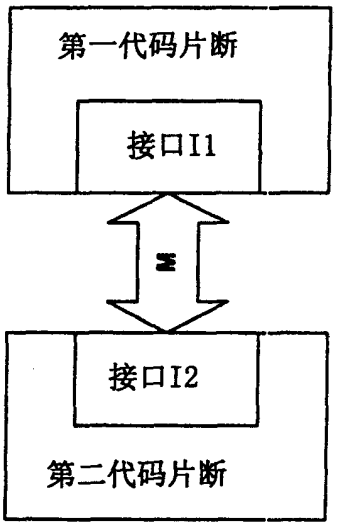

第一代码片断
接口I1
接口I2
第二代码片断

图 6

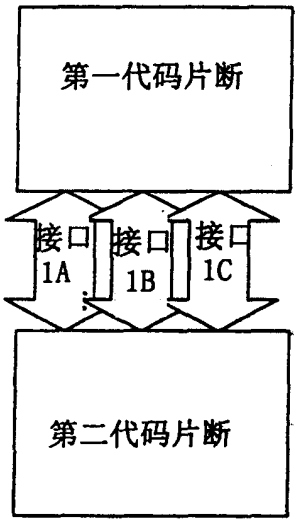

第一代码片断
接口1A
接口1B
接口1C
第二代码片断

图 7

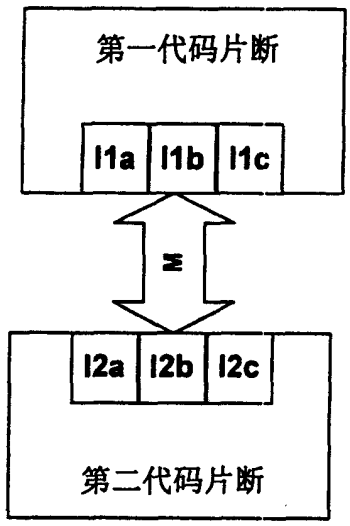

第一代码片断
I1a
I1b
I1c
I2a
I2b
I2c
第二代码片断

图 8

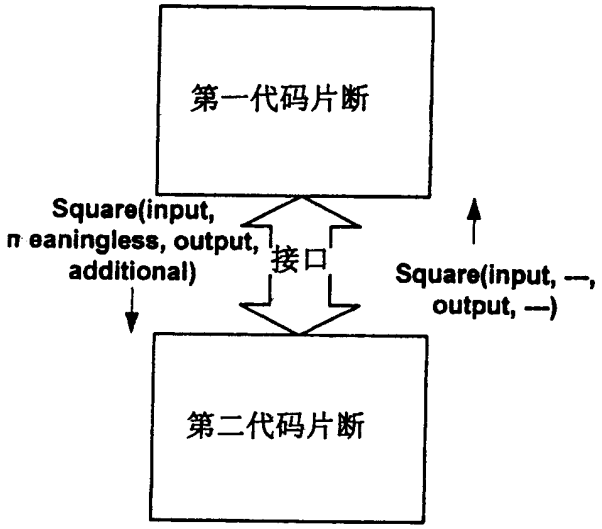


图 9

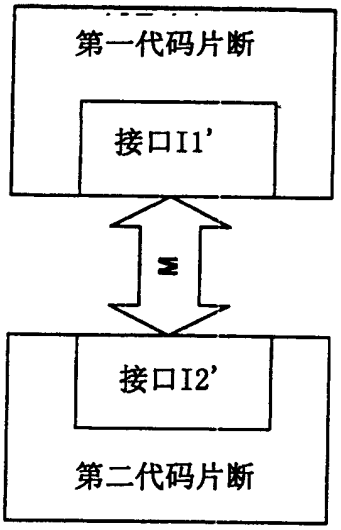


图 10

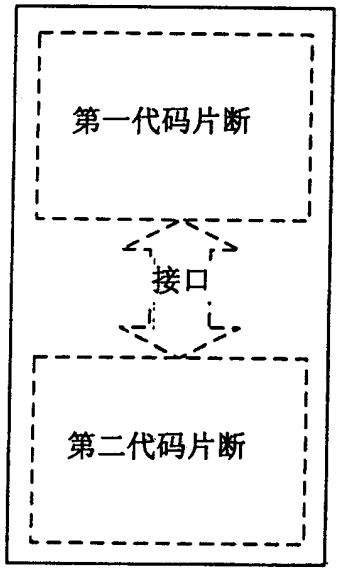


图 11

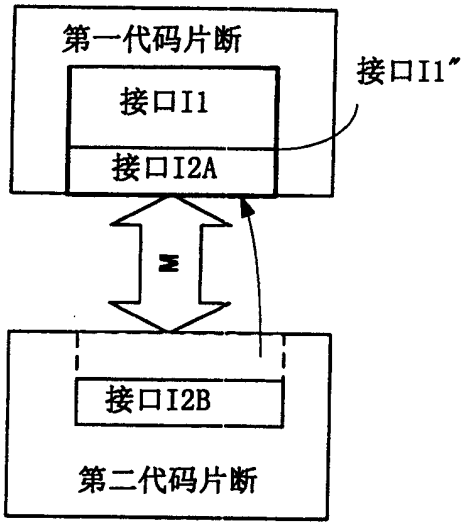


图 12

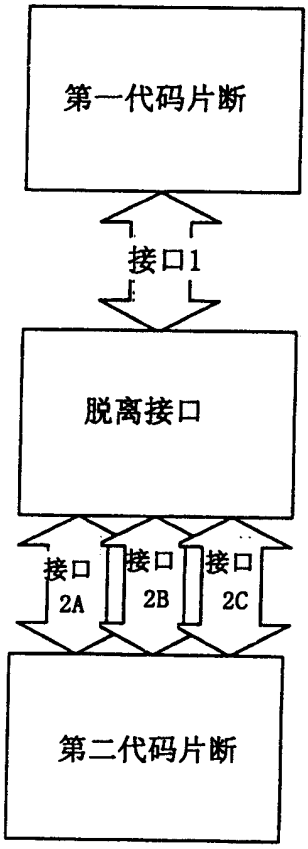

第一代码片断
接口1
脱离接口
接口 2A
接口 2B
接口 2C
第二代码片断

图 13

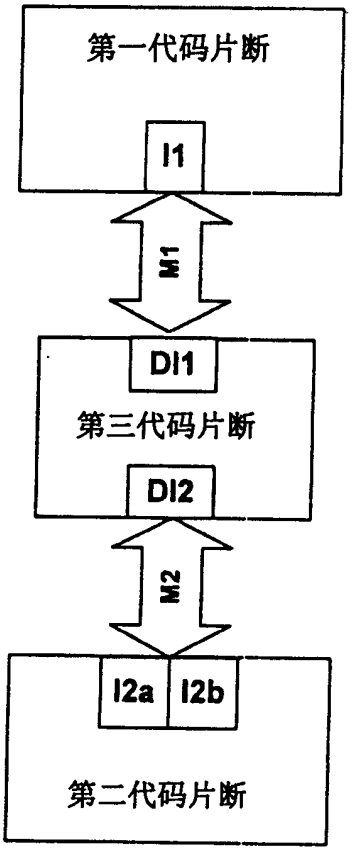

第一代码片断
I1
M1
DI1
第三代码片断
DI2
M2
I2a
I2b
第二代码片断

图 14

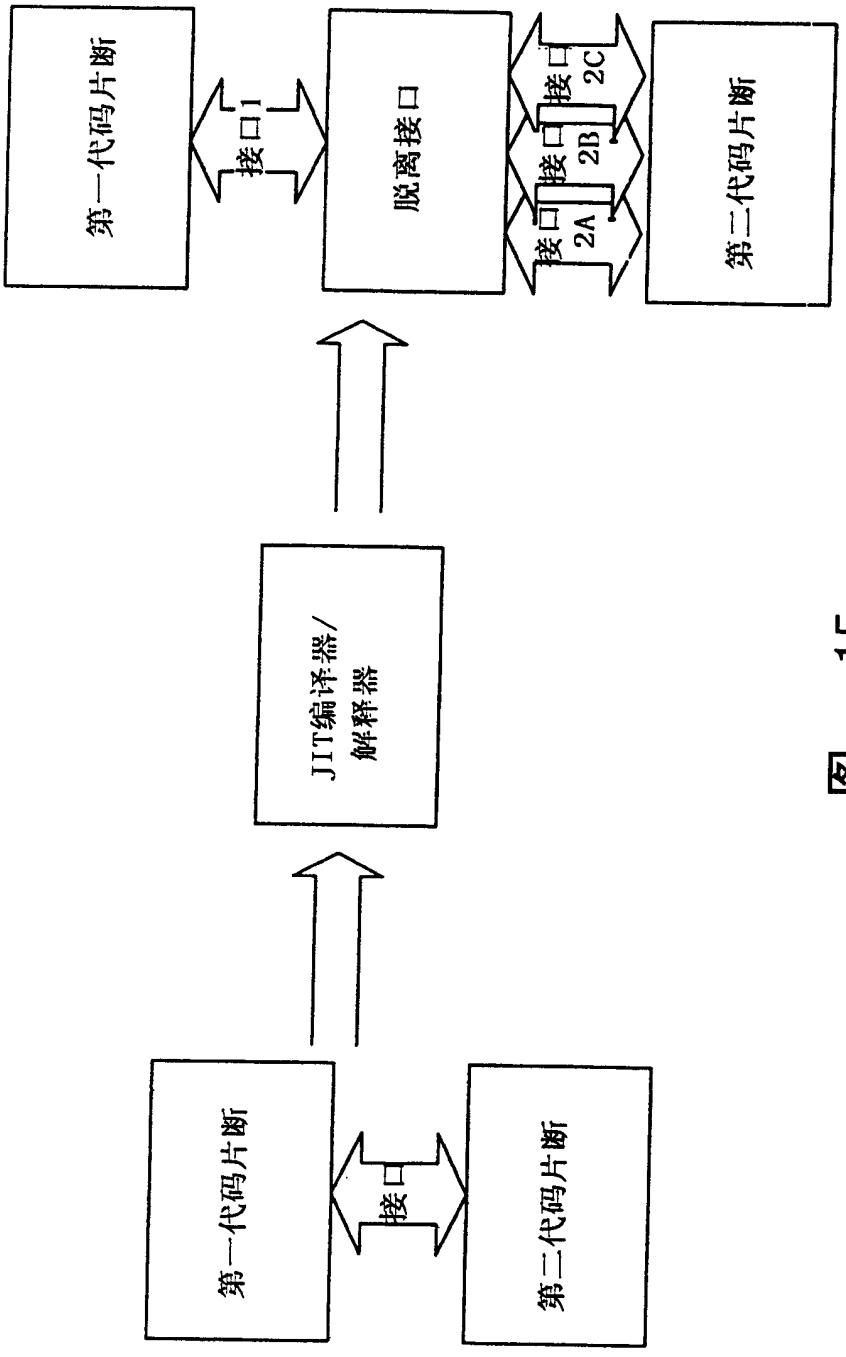

第一代码片断
接口1
脱离接口
接口2C
接口2B
接口2A
第二代码片断
JIT编译器/解释器
第一代码片断
接口
第二代码片断

图 15

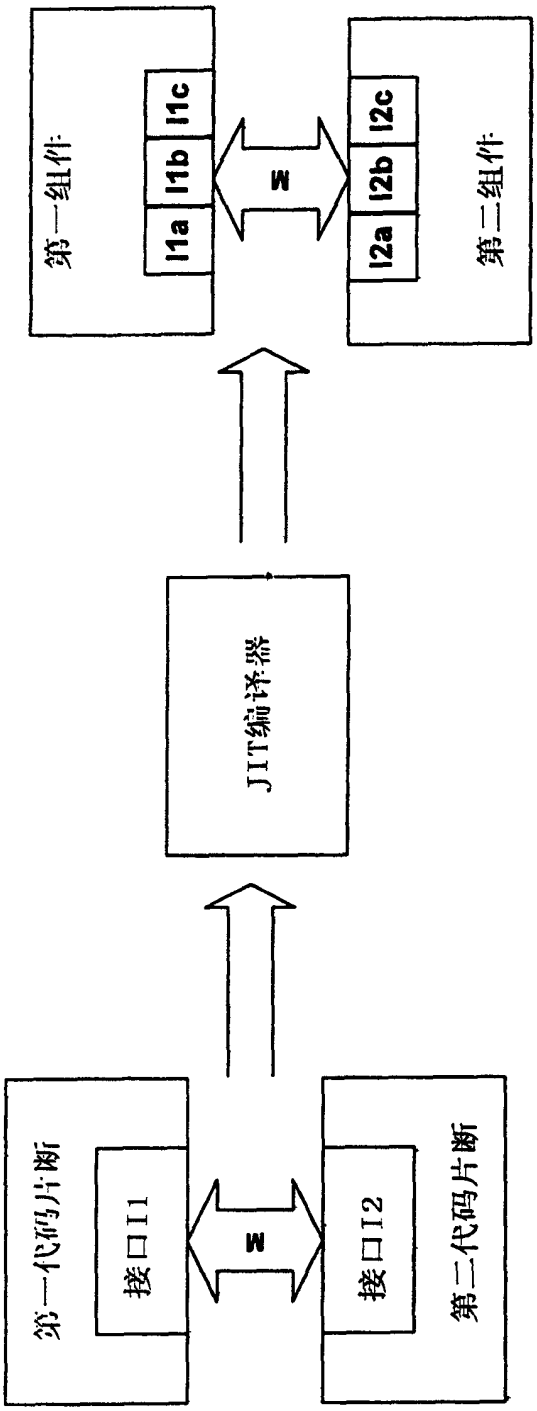

第一组件
I1a
I1b
I1c
第二组件
I2a
I2b
I2c
JIT编译器
第一代码片断
接口I1
第二代码片断
接口I2

图 16