

[19] 中华人民共和国国家知识产权局

[51] Int. Cl.

G06F 3/00 (2006.01)

G06F 13/00 (2006.01)



[12] 发明专利说明书

专利号 ZL 99106287.6

[45] 授权公告日 2006 年 10 月 18 日

[11] 授权公告号 CN 1280689C

[22] 申请日 1999.3.22 [21] 申请号 99106287.6

[30] 优先权

[32] 1998.3.20 [33] US [31] 044918

[71] 专利权人 太阳微系统公司

地址 美国加利福尼亚

[72] 发明人 卡普诺·D·卡特

审查员 杨蕊

[74] 专利代理机构 中国国际贸易促进委员会专利
商标事务所

代理人 于静

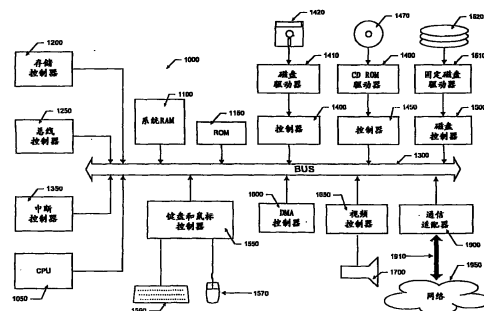
权利要求书 2 页 说明书 6 页 附图 9 页

[54] 发明名称

动态配置图形用户界面的方法

[57] 摘要

一种系统动态构造出一种与一应用程序有关的图形用户界面。一命令类库存储了表示与所述应用程序结合的附加功能的命令类。一执行单元设法在所述应用程序的运行期例示命令类，确定哪个命令类被成功例示，并构造出用于该应用程序的图形用户界面以使所述图形用户界面含有成功例示的命令类的附加功能。



1.一种计算机执行方法，用于动态构造出一与一程序有关、与所述程序有关的命令类有关的用户界面，所述命令类表示所述程序所用的附加功能，该方法包括步骤：

尝试不更新所述程序而例示所述命令类；

确定哪个命令类被成功例示；及

构造所述用户界面，以便用户界面含有对应于成功例示的命令类的附加功能；

在无需用户介入下，通过插入用于调用成功例示的命令类的附加功能的直观控件修改所述用户界面；以及

在直观显示器上显示修改后的图形用户接口。

2.根据权利要求1所述的方法，其特征在于，所述尝试性步骤包括子步骤：

定位命令类的成功例示的数据项，以及在所有数据项被定位时，声明例示成功。

3.根据权利要求2所述的方法，其特征在于，所述声明步骤包括子步骤：

在定位子步骤不能定位至少一个数据项时，确定例示失败。

4.根据权利要求1所述的方法，其特征在于，所述构造步骤包括子步骤：

通过增加或删除用于调用成功例示的命令类的附加功能的直观显示来修改用户界面。

5.根据权利要求1所述的方法，其特征在于，所述构造步骤包括子步骤：

当尝试步骤不能成功地例示任何命令类时，在一直观显示器上显示一基本用户界面。

6.一种动态构造与程序有关的图形用户界面的方法，其包括以下步骤：

在一存储器内存储多个命令类，每一命令类均表示用于程序的一模块；

执行所述程序；

设法在程序执行过程中在不必更新程序而定位用于命令类的成功例示的数据项，

只有命令类的所有数据项均被定位时，才声明一个命令类的成功例示；

无需用户介入下通过增加或删除用于调用与成功例示的命令类有关的模块的直观控件构造出所述图形用户界面，或形成用于所述程序的一基本图形用户界面。

动态配置图形用户界面的方法

本发明涉及软件程序设计系统，且更特别涉及这样的方法及系统，即基于运行期有效的附加特征动态重新配置与应用程序有关的一图形用户界面。

软件开发已认识到在计算机程序逻辑设计中，使用模块化概念的优点。使用了模块化概念的软件设置更便于其它硬件平台，更适于其它软件和硬件接口，并更易于设计，检测，安装和维护。软件开发将程序逻辑设计为能完成特定任务的分离软件模块(部件)组。开发者将每一模块均设计为尽可能独立和抽象于其它模块。

面向对象程序设计能促进模块软件的设计。面向对象程序设计允许开发者使一信息包与一组用于处理信息的方法(或功能)相联系。具有其相关方法的信息包统称为一个目标。通过访问一特别目标，开发者获得了对目标相关方法的隐式访问。

面向对象系统在对目标的描述和目标本身之间会产生差别。通过相同的广义描述可描述出不止一个相似的目标。由于该描述能被用于说明一种或一“类”相似的目标，所以目标描述被称为一个类。换句话说，一个目标即为一个类的一实例。使用目标和类的一种面向对象程序设计语言的一个实例为Sun Microsystems™ Java™面向对象程序设计语言，该语言在于1997年由Sams.net发表、题目为“Java 1.1 Developer's Guide”的文章中披露，其作者为Jamie Jaworski, Ed，其可在此处参考使用。Sun, Sun Microsystems, the Sun Logo, Java, 及以Java基础的商标为Sun Microsystems公司在美国或其它国家的商标或注册商标。

软件开发通常向用户提供运行期操作的应用程序所需的所有类。但是，一个应用程序可要求不在运行期出现的类。例如，可考虑将一计时模块与一邮件应用程序结合。计时模块对邮件应用程序提供附加功能，如允许约定及事件的调度。邮件应用程序的图形用户界面可包括一用于调用约定调度功能的调度约定按钮。但如果计时类不可用，则调度约定按钮是无用的。

软件开发通过创建两个分离版本的邮件应用程序：一个与计时器结合，而另一个则不与计时器结合来解决这一问题。以此方式，软件开发通过只提

供用户期望的那些附加功能就能满足用户的需求。

这种解决方法的问题在于要求软件开发者创建用户所需要的同一应用程序的具有不同等级附加功能的多个版本。然而，软件开发者可仅创建综合有所有附加功能的唯一版本应用程序。但是，这样会增加软件产品的成本并迫使仅希望得到基本应用程序的用户去购买那些附加功能。

因此，存在这样一种需求，即在不要求创建多版本应用程序的情况下，向用户提供一种仅具有期望功能的应用程序。

本发明的系统和方法通过将附加功能移至可与基本应用程序分开出售的独立类或模块，在运行期确定可用的功能，并修改与该应用程序有关的图形用户界面以显示对可用功能的控制来实现这一需求。

本发明的一种系统动态地构造出一种与一应用程序有关的图形用户界面。一命令类库存储了表示与所述应用程序结合的附加功能的命令类。一执行单元设法在所述应用程序的运行期例示命令类，确定已被成功例示的命令类，并构造出图形用户界面以使所述图形用户界面含有被成功例示的命令类的附加功能。

作为本发明说明书的一部分的附图描述了本发明的一个实施例，并与说明书一起说明了本发明的目的，优点和原理，在这些附图中：

图1为本发明的计算机系统的示意图；

图2为本发明实施方案中，图1所示的一存储器的示意图。

图3为本发明实施方案中，图2所示的命令类库的示意图；

图4A, 4B和4C为本发明的运行期运行的流程图；及

图5A, 5B和5C说明了一种用于根据本发明构成的邮件应用程序的图形用户界面的实施例。

本发明的详细描述应参照附图。在不同附图中的相同标号表示相同或相似的部件。同样，以下的详细描述不应构成对本发明的限制。而本发明的保护范围应由后述的权利要求限定。

本发明的系统和方法能根据在运行期被确定为可用的附加特征重新来配置与一程序有关的图形用户界面。即，在执行期间，所述程序确定了哪些类是可用的，并随后构造一图形用户界面以反映出所有的可用类。每一类通常表示由程序提供的一个特征。例如，用于一邮件程序的一个类表示了一个计时功能。

图1为用于本发明可实现的计算机系统的系统体系结构的示意图。图1的计

计算机系统示例仅出于对发明进行说明的目的。虽然该描述可参照用于说明特定计算机系统，如IBM PS/2个人电脑中常用的术语，但这些描述和概念同样适用于其它的计算机系统，如网络计算机，工作站，甚至具有不同于图1所示的结构的大型计算机。

此外，虽然本发明同样适用于具有相同要求的其它计算机系统，但此处是参照执行Java程序语言，Java™运行期环境，及Java™虚拟机规范的计算机系统对该设备进行描述的。具体地，本发明可实现面向对象和非面向对象的程序设计系统。

计算机系统1000包括一个可采用传统微处理器的中央处理单元(CPU)1050，一个用于临时存储信息的随机存取存储器(RAM)1100及一个用于永久存储信息的只读存储器(ROM)1150。一个存储控制器1200用于控制RAM1100。

一总线1300用于连接计算机系统1000中的各部件。一总线控制器1250用于控制总线1300。一中断控制器1350用于接收并处理来自系统各部件的各种中断信号。

海量存储器由磁盘1420，CD ROM1470，或硬盘驱动器1520构成。数据和软件通过可卸接的媒体，如磁盘1420和CD ROM1470与计算机系统1000进行交换。磁盘1420可插入磁盘驱动器1410，而该磁盘驱动器本身通过一控制器1400依次与总线1300相连。同样，CD ROM1470可插入CD ROM驱动器1460，该CD ROM驱动器通过一控制器1450依次与总线1300相连。硬盘1520构成了固定的磁盘驱动器1510的一部分，该固定磁盘驱动器本身通过一个控制器1500与总线1300相连。

可通过多个装置对用户提供的计算机系统1000的输入。例如，通过控制器1550使键盘1560和鼠标1570与总线1300相连。显然，对于本领域有经验的技术人员来说，还可根据需要使其它输入装置，如输入笔，输入板或语言识别装置与总线1300及一个适合的控制器和软件相连。DMA控制器1600用于执行对RAM1100的直接存储器存取。通过控制视频显示器1700的视频控制器1650可产生用户输出信息。

计算机系统1000还包括一通信适配器1900，其允许所述计算机系统通过图中由总线1910和网络1950所示的一局域网(LAN)或一广域网(WAN)，如Internet与附加计算资源互连。通过网络1950传送的信号通常被称为“载波”，其能够传送信息。虽然本发明特征被描述为是以指令形式被存入存储器的，

但本领域有经验的技术人员也应理解那些技术特征也可存储在其它的计算机可读媒体上, 如象硬盘、软盘, 或CD-ROM这样的次级存储器, 或RAM或ROM的其它形式, 或一载波。

计算机系统1000的运行通常由操作系统软件控制和协调。所述操作系统用于控制系统资源的分配并执行其它事务中, 诸如存储管理, 进程调度, 联网及服务之类的任务。

图2为本发明中存储器2000的示意图。存储器2000可包括系统RAM1100(图1), 但也可包括其它类型的存储器, 这些存储器可包括图1所示的任意一种存储器。

存储器2000包括运行期系统2100, 应用程序2200, 及数据存储器 2300。Java虚拟机(JVM)2120处于Java运行期系统2100的核心。JVM2120为一在软件中实现的微处理器, 所述软件可利用由操作系统和计算机硬件提供的功能而进行运行。此处, JVM如例如于1996年由Tim Lindholm及Frank Yellin, Addison Wesley发表的、题目为“The Java Virtual Machine Specitication”的文章所述的JVM, 本文将其引入作为参考。运行期系统2100还包括命令类库2140。命令类库2140将参照附图3在下文中加以描述。

应用程序2200包括由CPU1050执行并由JVM2120解释的程序。所述程序包括面向对象的程序, 如采用Java程序语言的小程序, 以及其它面向对象的程序, 多平台程序, 保密程序, 分布式网络程序, 多线程程序, 网络程序等, 其中的一些程序在存储在数据存储器2300中的目标上进行操作。数据存储器2300存储了CPU1050和JVM2120所使用的数据。

命令类库示例

图3为命令类库2140的示意图。命令类库2140存储了多个命令类3100-3400, 若被成功例示, 则这些命令类包括用于一应用程序的附加功能。例如, 命令类3100包括构造程序3120和程序包部分3140。构造程序3120记录了命令类3100所依赖的(即要求执行其功能的其它类)的外部程序包(或数据项)的等同列表。程序包部分3140用于包括例示这些程序包的Java编程语言的常规代码。

当由JVM2120调用时, 构造程序3120设法定位命令类3100被成功例示所需的一些程序包(或一个程序包中的类)。构造程序3120不必例示所找到的程序包, 而只需确定它们是否存在。在应用程序需要的情况下, 程序包部分3140稍后以常规方式例示这些程序包。

如果构造程序3120不能确定一程序包，则构造程序3120发出异常信号。该异常信号通知JVM2120命令类3100的例示失败，因此由命令类提供的特征对应用程序用户是不可用的，从而不能包含在图形用户界面中。

系统操作

图4A - 4C为本发明运行期运行的流程图。图5A - 5C说明了一个用于依据本发明而构成的邮件应用程序的图形用户界面实施例。

在执行如Java编程语言这样的面向对象程序设计语言的过程中，JVM2120执行多项操作，包括应用程序的执行[步骤4110]。当JVM2120被描述为执行应用程序时，这一过程实际上由CPU1050通过Java运行期系统2100完成。

在应用程序执行期间的某一点上，JVM2120会遇到用于构造图形用户界面的指令，该图形用户界面允许用户与所述应用程序相互联系[步骤4120]。根据遇到的这些指令，JVM2120访问命令类库2140，以便设法例示含有应用程序可用的附加功能的命令类。JVM2120利用一“try-catch”子句调用命令类3100中的构造程序3120(图3)[步骤4130]。try-catch子句允许应用程序设法例示一命令类并在由于该命令类不能被例示而发出异常信号的情况下仍继续运行。

一旦被调用，构造程序3120设法从列表构造程序3120中定位使命令类3100被成功例示所需的程序包(类)[步骤4140]。对于每一个被成功定位的程序包，构造程序3120除放弃外，使一指针返回程序包。如果不能定位一程序包，则对于一不可用的计时命令类，构造程序3120发出异常信号，如：

```
public Calendar Command throws ClassNotFoundException  
{if(getClass.getResource("/sunw/calender/Calender.class")==null)th  
row new ClassNotFoundException()};
```

如果不能定位一个所需的程序包，构造程序3120声明例示失败[步骤4150]，并以发出异常信号的方式向JVM2120报告例示失败[步骤4210](图4B)。随后，JVM 2120确定是否存在任何需例示的剩余命令类[步骤4220]。如果有一个命令类存在，JVM2120返回步骤4130(图4A)以为该命令类调用构造程序。

如果不存在需例示的剩余命令类，JVM2120则确定是否任意的命令类均被成功例示[步骤4230]。如果没有命令类被例示，随后，JVM 2120仅通过应用程序正确操作所需的基本控件在视频显示器(图1)上显示一基本(或隐含)图形用户界面[步骤4240]。图5A说明了一用于邮件应用程序、具有基本控制件的图形

用户界面的实施例。图形用户界面包括四个按钮式基本控件：In Box5110, Out Box5120, Trash5130, 及Send Mail5140。这些按钮能调用本领域熟知的功能。

但是，如果已成功地例示了一个或多个命令类，JVM2120则通过与成功例示的命令类有关的附加控件在视频显示器1700上显示图形用户界面[步骤4250]。图5B描述了用于邮件应用程序、具有附加控件的一图形用户界面实施例。该图形用户界面包括四个基本控件5210 - 5240及用作拼写检查程序5250和词库5260的附加控件。这些按钮能调用本领域熟知的功能。

如果在步骤4150(图4A)中，构造程序3120能成功定位所有的必需程序包，则构造程序3120声明例示成功，并向JVM2120报告例示成功[步骤4310](图4C)。作为响应，JVM2120重新配置图形用户界面以增加一个或多个用于成功例示的命令类的控件[步骤4320]。如上所述，所述控件使命令类所提供的服务可用用户。

随后，JVM2120判断是否存在任意需被例示的剩余命令类[步骤4330]。如果存在一个剩余命令类，JVM2120返回步骤4130(图4A)以为该命令类调用构造程序。但是，如果不存在被例示的剩余命令类，则JVM2120通过与被成功例示的命令类有关的附加控件，在视频显示器1700上显示图形用户界面[步骤4340]。图5C描述了用于邮件应用程序具有附加控件的图形用户界面的一个实施例。该图形用户界面包括四个基本控件5310 - 5340及用作计时器5350，拼法检查程序5360及词库5370的附加控件。这些按钮能调用本领域熟知的功能。

本发明的系统和方法可根据运行期确定的可用特征来动态确定与应用程序有关的图形用户界面的形式。

前面对本发明最佳实施例的描述仅是描述和说明，但前面的描述并非包括一切或不应构成对所披露的本发明正确格式的限定。因此，根据上面的教导或从本发明的实践出发可对本发明作出多种修改和改进。本发明的保护范围应由权利要求和其等同物限定。

例如，本发明的实施方案已被描述为以一基本图形用户界面开始并根据在运行期发现可用的附加特性增加直观控件。但是，实际情况不限于此。图形用户界面初始时可加载所有控件。随后，根据在运行期确定可用的附加特性，从所述界面删除控件。

另外，虽然前面的描述说明了一图形用户界面的产生，但这些描述和概念同样还适于其它型的用户界面，例如基于文本的用户接口。

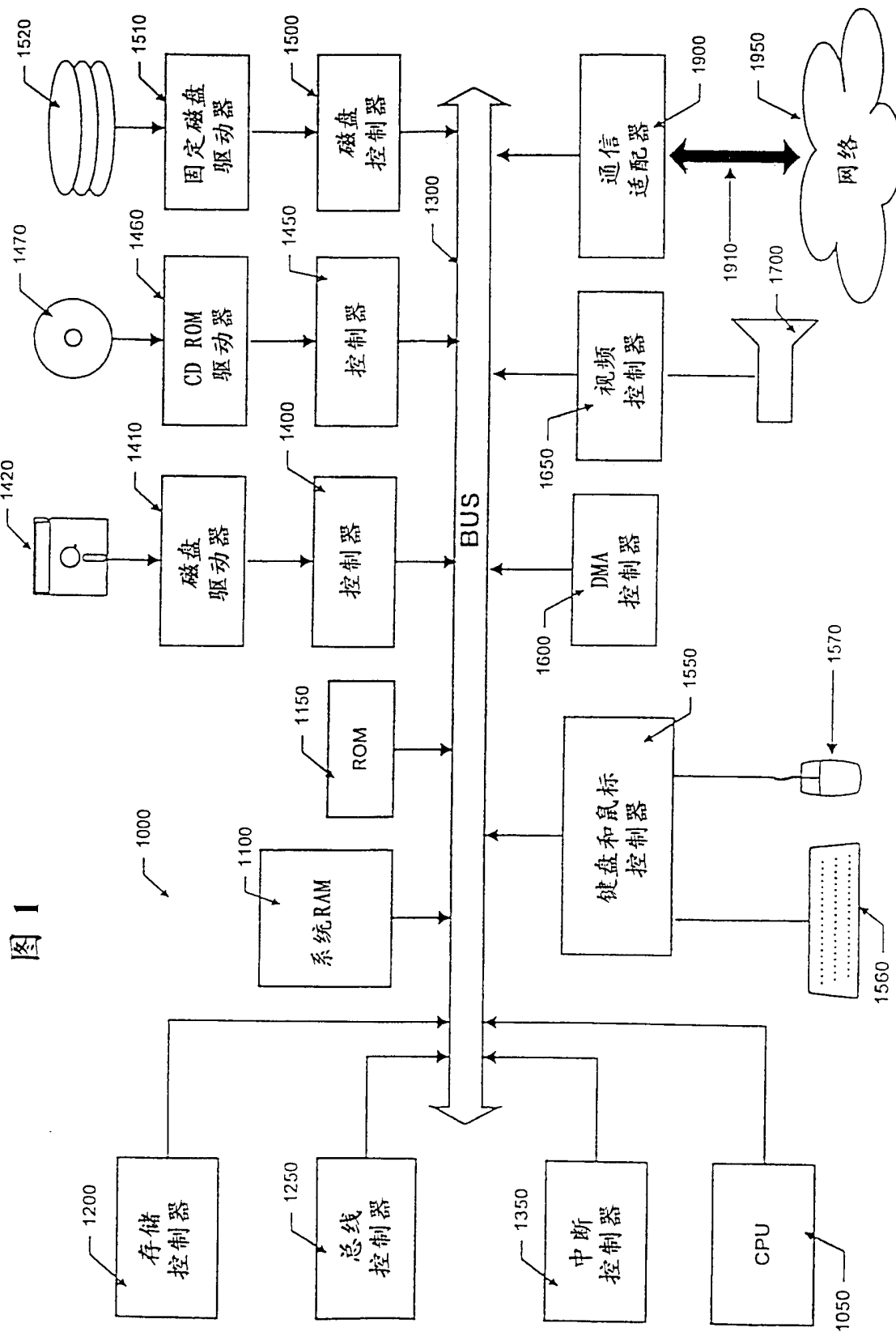


图 2

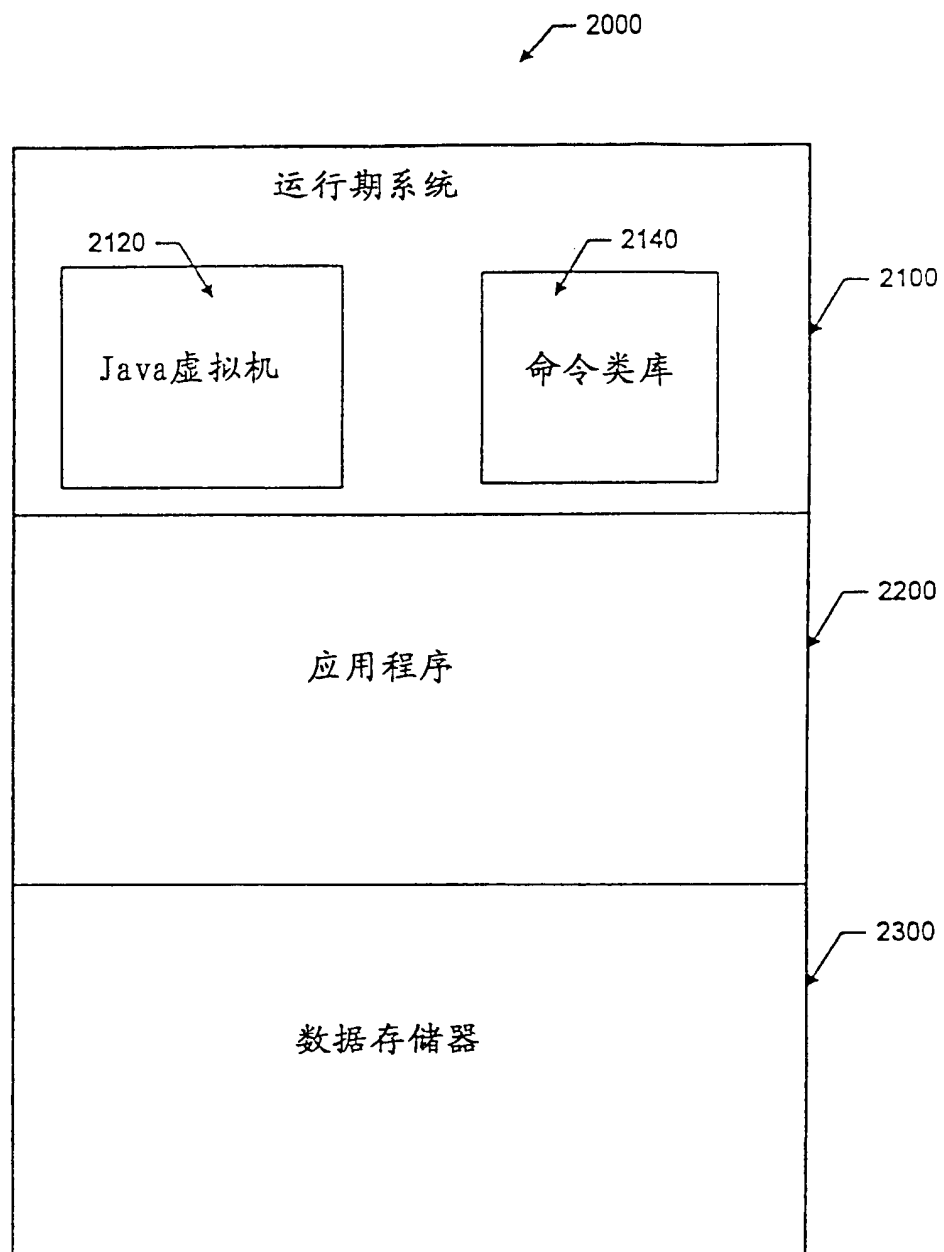


图 3

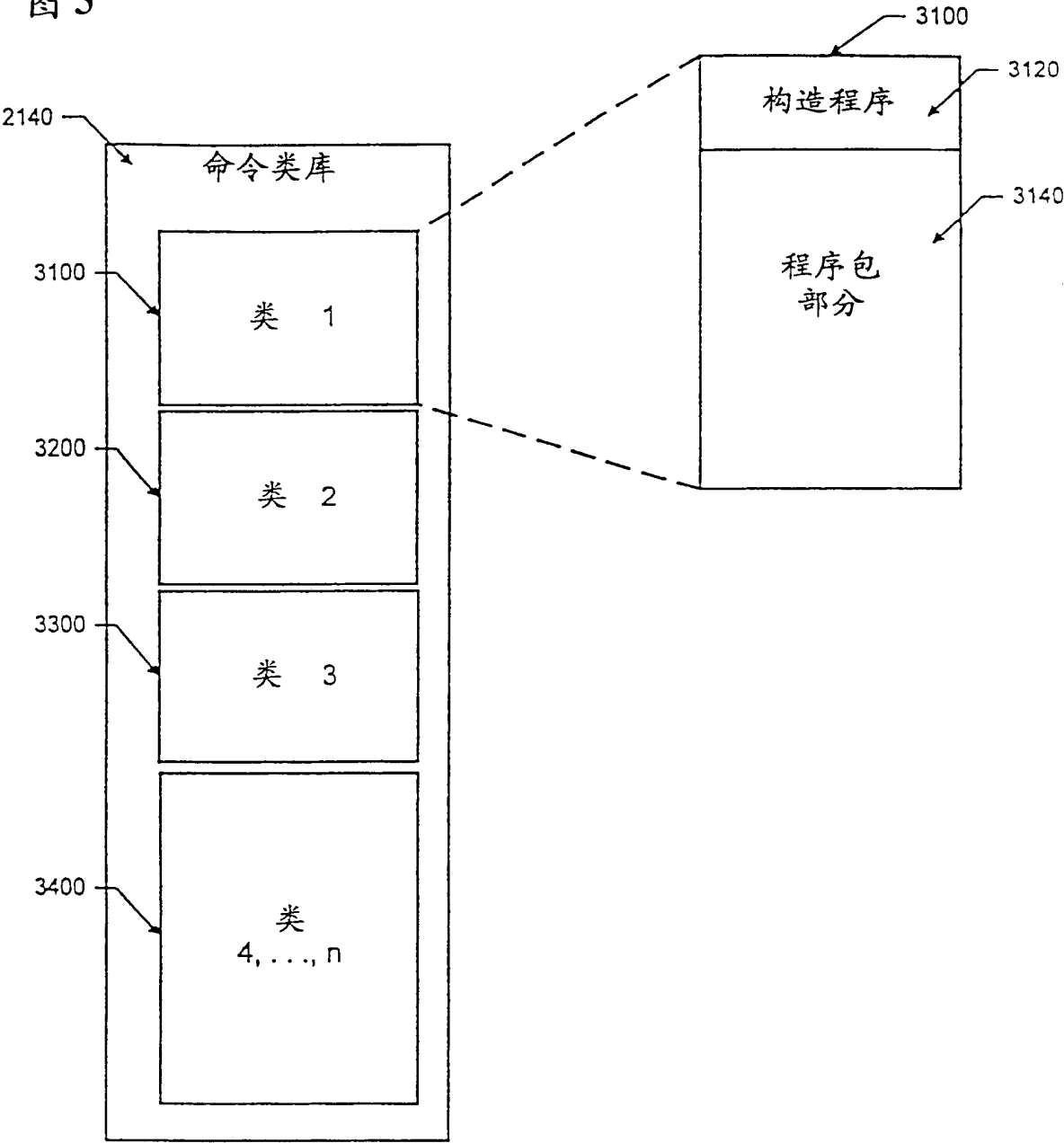


图 4A

运行期过程

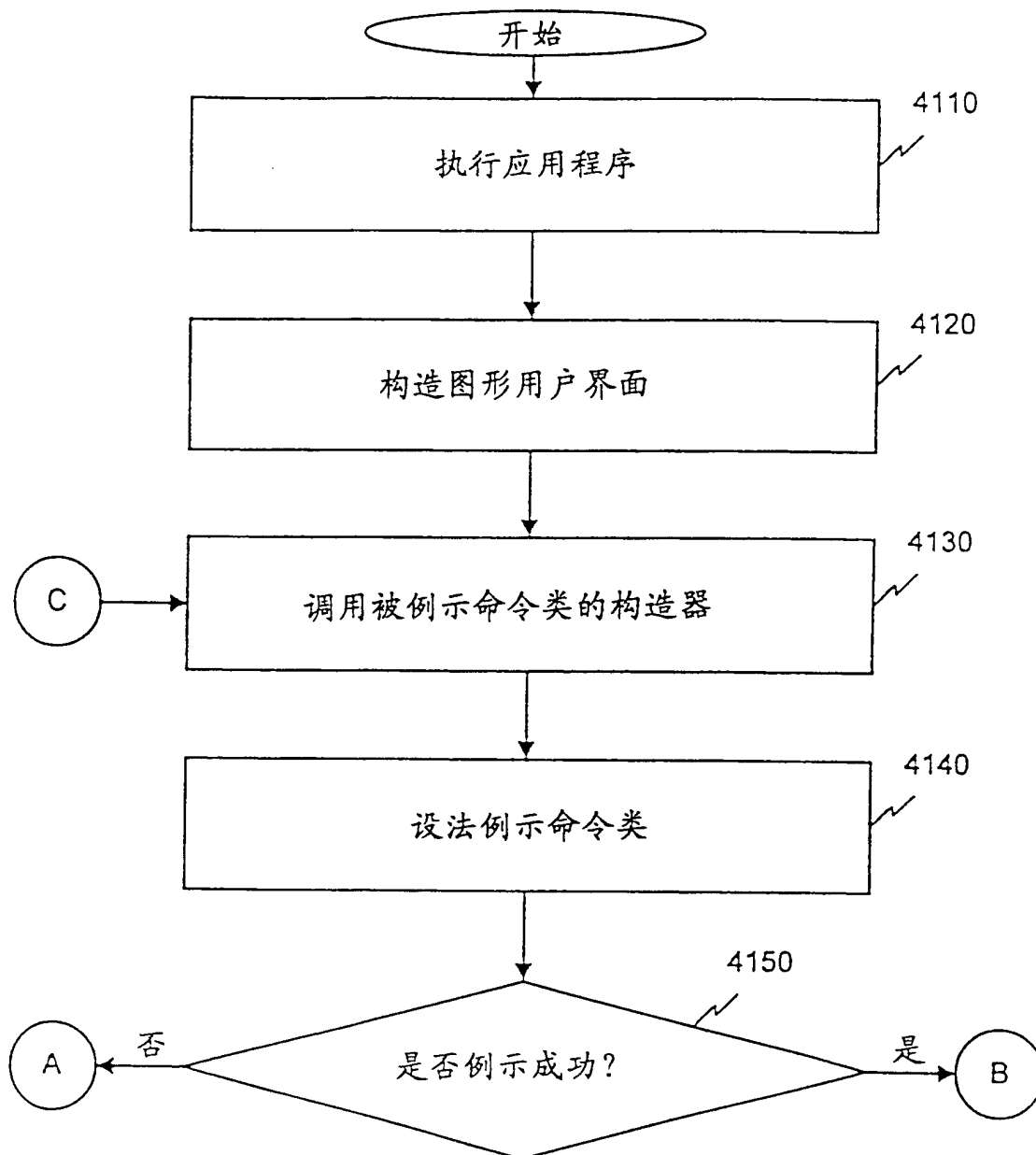


图 4B

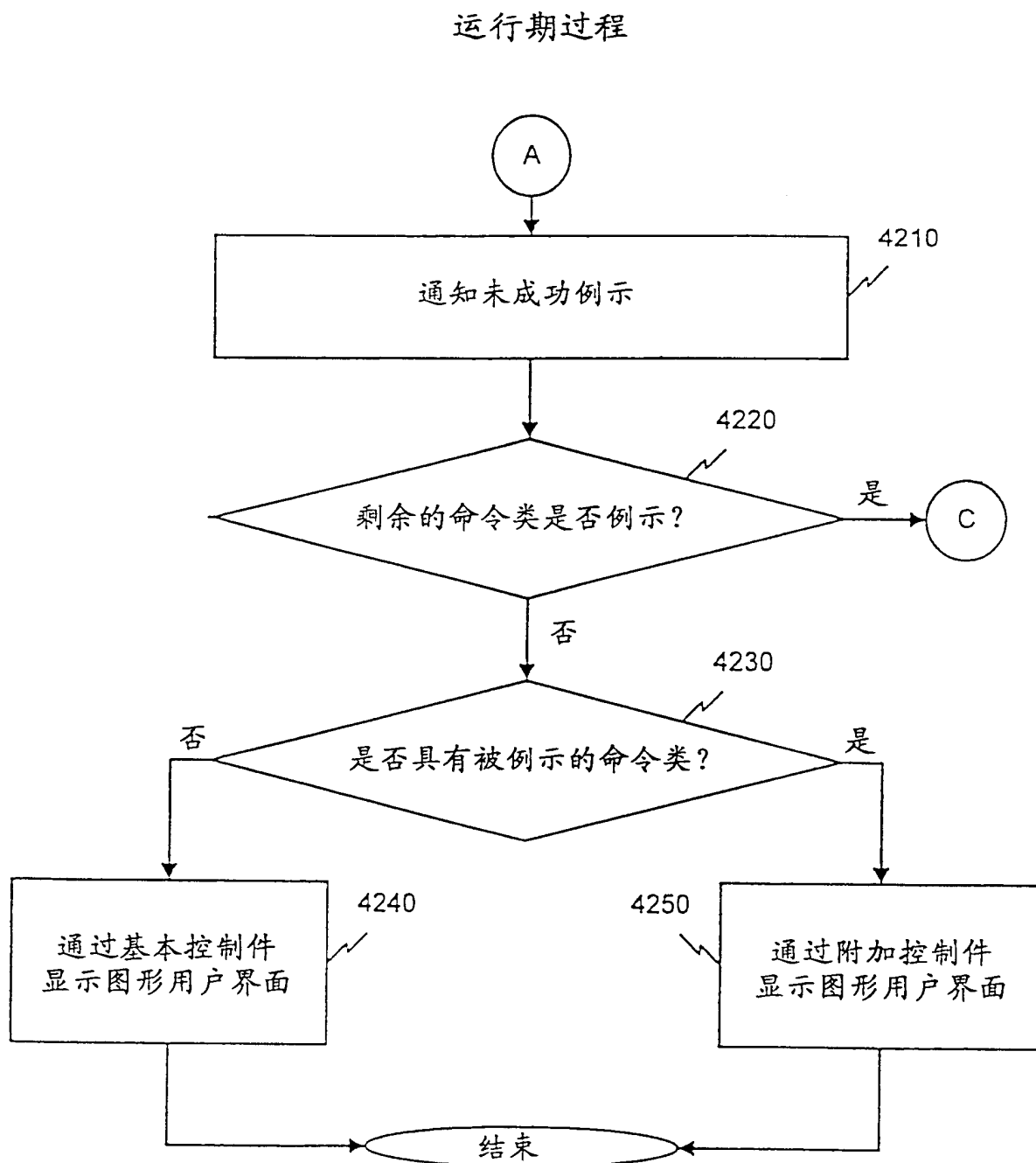


图 4C 运行期过程

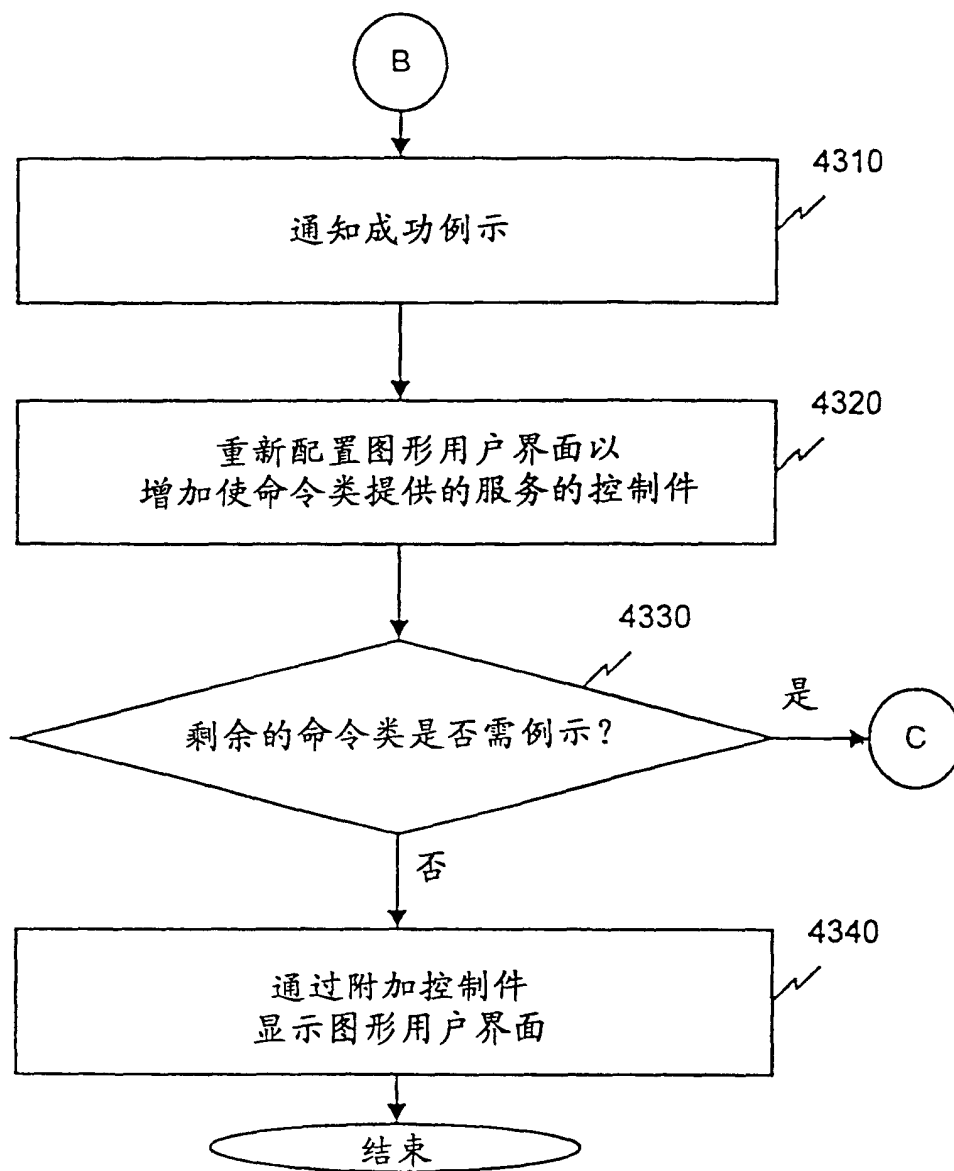


图 5A

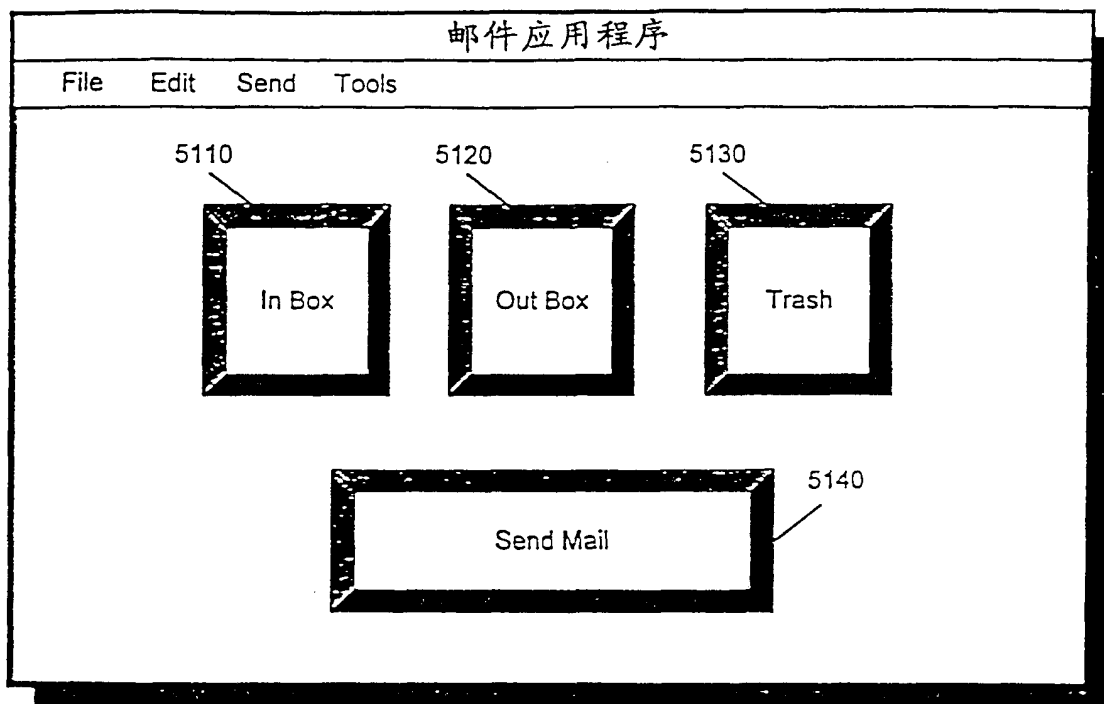


图 5B

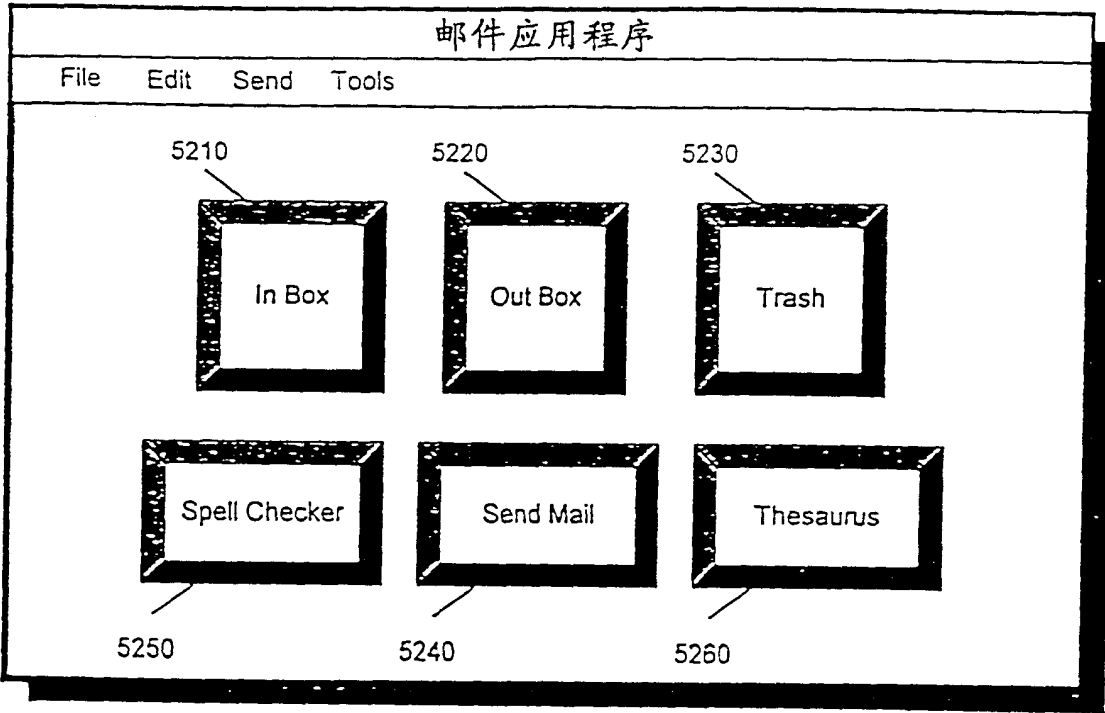


图 5C

