



[12] 发明专利说明书

专利号 ZL 200510052450.6

[45] 授权公告日 2008 年 12 月 17 日

[11] 授权公告号 CN 100444108C

[22] 申请日 2005.2.28

[21] 申请号 200510052450.6

[30] 优先权

[32] 2004.2.27 [33] EP [31] 04251158.4

[73] 专利权人 捷讯研究有限公司

地址 加拿大安大略省沃特卢市

[72] 发明人 卡门·B·维塔诺夫

迈克尔·申菲尔德

布伦杜沙·L·弗里奇

[56] 参考文献

US5596702A 1997.1.21

US5991762A 1999.11.23

US6448981B1 2002.9.10

审查员 刘邵频

[74] 专利代理机构 中科专利商标代理有限责任公

司

代理人 王 玮

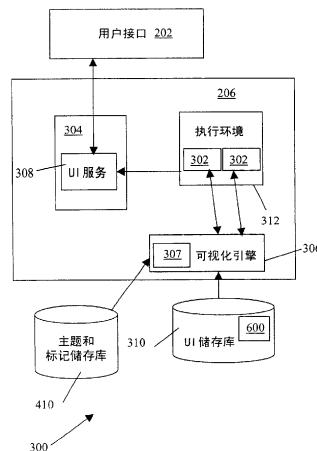
权利要求书 5 页 说明书 16 页 附图 6 页

[54] 发明名称

使用 UI 储存库的公共 UI 组件执行无线应用程序的系统和方法

[57] 摘要

本发明提供了产生用于显示在无线设备的用户接口 (UI) 上的屏幕表示的方法，将屏幕表示定义为以结构定义语言来表达的 UI 定义集合，结构定义语言被配置为由被配置为在无线设备上运行的多个应用程序所引用，包括：接收来自多个应用程序中的第一应用程序的对屏幕表示的请求；从无线设备上的存储器中检索与屏幕表示相对应的 UI 定义集合；解析 UI 定义的结构定义语言，以确定屏幕表示的功能特征；将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；以及将屏幕模型提供给第一应用程序，以便以当前 UI 显示条件和当前屏幕值进行填充，以产生屏幕表示；其中将屏幕表示配置用于随后对 UI 的显示，以通过用户事件与用户进行交互。本发明还提供了相应系统。



1、一种用于产生用于显示在无线设备的用户接口上的屏幕表示的方法，将屏幕表示定义为以结构定义语言来表达的用户接口定义集合，结构定义语言被配置为由被配置为在无线设备上运行的多个应用程序所引用，所述方法的特征在于包括以下步骤：

接收来自多个应用程序中的第一应用程序的对屏幕表示的请求；

从无线设备上的存储器中检索与屏幕表示相对应的用户接口定义集合；

解析用户接口定义的结构定义语言，以确定屏幕表示的功能特征；

将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；以及

将屏幕模型提供给第一应用程序，以便以当前用户接口显示条件和当前屏幕值进行填充，以产生屏幕表示；

其中将屏幕表示配置用于随后对用户接口的显示，以便通过用户事件与用户进行交互。

2、根据权利要求1所述的方法，其特征在于将屏幕模型设置在由设备的应用程序执行环境所采用的平台相关或无关配置中。

3、根据权利要求2所述的方法，其特征在于第一应用程序是配置用于按照执行环境的浏览器适用形式执行的浏览器应用程序。

4、根据权利要求2所述的方法，其特征在于第一应用程序是配置用于按照执行环境的本地形式执行的本地应用程序。

5、根据权利要求2所述的方法，其特征在于第一应用程序是配置用于按照执行环境的应用程序容器形式执行的、基于定义计划的组件应用程序。

6、根据权利要求2所述的方法，其特征在于还包括以下步骤：从存储器的外观储存库中检索外观特征，所述储存库包括外观特征的外观再现规则。

7、根据权利要求 6 所述的方法，其特征在于从包括背景主题、标记颜色方案和标记布置的组中来选择规则。

8、根据权利要求 2 所述的方法，其特征在于还包括：从存储器的用户接口定义储存库中检索用户接口定义，所述用户接口定义储存库包括针对功能特性的用户接口定义。

9、根据权利要求 8 所述的方法，其特征在于还包括以下步骤：包括针对由用户接口定义解析而来的屏幕表示的属性的用户接口元素。

10、根据权利要求 9 所述的方法，其特征在于从包括逻辑名称、标题和缺省字体的组中来选择属性。

11、根据权利要求 10 所述的方法，其特征在于还包括以下步骤：包括针对规定了用户事件的处理的事件处理定义的用户接口元素，由用户接口定义解析出所述事件处理定义。

12、根据权利要求 8 所述的方法，其特征在于还包括以下步骤：包括针对规定了由用户接口定义解析而来的菜单项集合的屏幕菜单的用户接口元素。

13、根据权利要求 8 所述的方法，其特征在于还包括以下步骤：包括针对用于适应用户事件的用户接口控制的用户接口元素，由用户接口定义解析出所述用户接口控制。

14、根据权利要求 13 所述的方法，其特征在于还包括以下步骤：包括针对用于定义用户接口控制的次序和位置的用户接口布局的用户接口元素，由用户接口定义解析出所述用户接口布局。

15、根据权利要求 2 所述的方法，其特征在于还包括以下步骤：用户接口服务解释用户事件，并将其转发给第一应用程序，应用程序处理用户事件，并返回对用户接口服务的控制。

16、根据权利要求 15 所述的方法，其特征在于还包括以下步骤：通过修改相应的屏幕模型，响应用户事件，更新屏幕表示。

17、根据权利要求 12 所述的方法，其特征在于由从包括可视化引擎、第一应用程序、用户接口服务的组中所选择的实体来执行屏幕模型的填充。

18、根据权利要求 2 所述的方法，其特征在于第一应用程序实例化属于多个应用程序中的第二应用程序的用户接口定义集合，即作为存储器中的条目而被引用的用户接口定义集合。

19、根据权利要求 18 所述的方法，其特征在于通过与多个应用程序之一相对应的惟一应用程序标识符来链接存储器中的每个用户接口定义集合。

20、根据权利要求 18 所述的方法，其特征在于第二应用程序提供与用户接口定义相关的用户事件的事件处理。

21、一种用于产生用于显示在无线设备的用户接口上的屏幕表示的系统，将屏幕表示定义为以结构定义语言来表达的用户接口定义集合，结构定义语言被配置为由被配置为在无线设备上运行的多个应用程序所引用，所述系统的特征在于包括：

存储器，用于存储由多个应用程序进行引用的多个用户接口定义集合；

可视化引擎，用于接受多个应用程序中的第一应用程序的屏幕表示请求，以及用于解析从存储器中检索出的选定用户接口定义集合的结构定义语言，以确定屏幕表示的功能特征，选定的用户接口定义对应于所请求的屏幕表示；

与可视化引擎相连的屏幕模块，用于将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；

执行环境，用于向第一应用程序提供屏幕模型，以便以当前用户接口显示条件和当前屏幕值进行填充；以及

用户接口服务，用于再现屏幕模型，以便向用户接口提供屏幕表示；

其中无线设备的用户与用户接口上的屏幕表示进行交互。

22、根据权利要求 21 所述的系统，其特征在于将屏幕模型设置在由设备的应用程序执行环境所采用的平台相关或无关配置中。

23、根据权利要求 22 所述的系统，其特征在于第一应用程序是配置用于按照执行环境的浏览器适用形式执行的浏览器应用程序。

24、根据权利要求 22 所述的系统，其特征在于第一应用程序是

配置用于按照执行环境的本地形式执行的本地应用程序。

25、根据权利要求 22 所述的系统，其特征在于第一应用程序是配置用于按照执行环境的应用程序容器形式执行的、基于定义计划的组件应用程序。

26、根据权利要求 22 所述的系统，其特征在于还包括存储器的外观储存库，所述储存库包括外观特征的外观再现规则。

27、根据权利要求 26 所述的系统，其特征在于从包括背景主题、标记颜色方案和标记布置的组中选择规则。

28、根据权利要求 22 所述的系统，其特征在于还包括存储器的用户接口定义储存库，所述用户接口定义储存库包括针对功能特性的用户接口定义。

29、根据权利要求 28 所述的系统，其特征在于包括针对由用户接口定义解析而来的屏幕表示的属性的用户接口元素。

30、根据权利要求 29 所述的系统，其特征在于从包括逻辑名称、标题和缺省字体的组中选择属性。

31、根据权利要求 30 所述的系统，其特征在于包括针对规定了用户事件的处理的事件处理定义的用户接口元素，由用户接口定义解析出所述事件处理定义。

32、根据权利要求 28 所述的系统，其特征在于包括针对规定了由用户接口定义解析而来的菜单项集合的屏幕菜单的用户接口元素。

33、根据权利要求 28 所述的系统，其特征在于包括针对用于适应用户事件的用户接口控制的用户接口元素，由用户接口定义解析出所述用户接口控制。

34、根据权利要求 33 所述的系统，其特征在于包括针对用于定义用户接口控制的次序和位置的用户接口布局的用户接口元素，由用户接口定义解析出所述用户接口布局。

35、根据权利要求 32 所述的系统，其特征在于还包括用户接口服务，用于解释用户事件，并将其转发给第一应用程序，应用程序处理用户事件，并返回对用户接口服务的控制。

36、根据权利要求 35 所述的系统，其特征在于通过修改相应的

屏幕模型，响应用户事件，更新屏幕表示。

37、根据权利要求 32 所述的系统，其特征在于由从包括可视化引擎、第一应用程序、用户接口服务的组中所选择的实体来执行屏幕模型的填充。

38、根据权利要求 22 所述的系统，其特征在于第一应用程序实例化属于多个应用程序中的第二应用程序的用户接口定义集合，即作为存储器中的条目而被引用的用户接口定义集合。

39、根据权利要求 38 所述的系统，其特征在于通过与多个应用程序之一相对应的惟一应用程序标识符来链接存储器中的每个用户接口定义集合。

40、根据权利要求 38 所述的系统，其特征在于第二应用程序提供与用户接口定义相关的用户事件的事件处理。

使用UI储存库的公共UI组件执行无线应用程序的系统和方法

技术领域

本申请大体上涉及应用程序在无线设备的用户接口上的表示。

背景技术

今天，所使用的终端设备的数量持续增加，如移动电话、具有无线通信能力的 PDA 和双向寻呼机等。运行在这些设备上的软件应用程序增加了其功能性。例如，移动电话可以包括检索城市范围的天气的应用程序，或者 PDA 可以包括允许用户购买杂货的应用程序。这些软件应用程序利用对网络的连接性，以便向用户提供及时有用的服务。但是，由于一些设备的受限资源，以及向设备传递大量数据的复杂性，开发针对多种设备的软件应用程序仍然是困难而耗时的任务。

目前，将设备配置为通过基于因特网的浏览器和/或本地应用程序与网络服务进行通信。本地应用程序具有针对设备平台的类型专门研发的优点，从而针对每次运行时刻的环境，提供了相对优化的应用程序。但是，本地应用程序具有以下缺点：非平台无关，必需开发相同应用程序的多个版本，以及尺寸相对较大，从而给设备的存储器资源造成了严重的负担。此外，应用程序开发商需要使用如 Java 和 C++ 等编程语言的经验，以便构建这些硬编码本地应用程序。存在对能够运行在客户端设备上的应用程序的需要，其具有广泛的运行时刻环境，以及具有对设备资源的减少的消耗。

需要使用动态且交互式的用户接口（UI），在定义用于管理设备（如无线等）上的应用程序表示的应用程序组件屏幕时，提供最大程度的灵活性和效率。由于无线设备资源的限制，重要的是具有一种使用减少的可执行代码的有效应用程序数据表示方法。

这里所公开的系统和方法提供了一种用于产生用户接口元素的

执行环境，以排除或消除上述缺点中的至少一些。

发明内容

需要的是利用动态、交互式用户接口（UI），在定义用于管理设备上的应用程序表示的无线应用程序的组件屏幕时，提供最大的自由度和效率。由于无线设备资源的限制，重要的是具有一种使用减少的可执行代码来进行有效应用程序数据表示的方法。与目前的用户接口可视化系统和方法相反，提供了一种系统和方法，具有智能运行时刻设备框架的执行环境，用于在设备上声明的用户接口（UI）上产生用户接口元素。所提出的方法通过 XML 元数据 UI 定义（或其他结构定义语言计划（schema））实现了用户接口定义，代替了需要在应用程序的执行代码中实现屏幕元素。将 UI 定义存储在公共 UI 储存库中，作为设备上的应用程序的公共资源，并在运行时执行。UI 定义与设备的目标平台无关。可以根据需要，定制和标记设备上的所有应用程序的“感观”。与应用程序逻辑分离地定义 UI 元素的布局和次序提供了应用程序的模块化。这种模块化允许重新使用已经定义的 UI 屏幕，并在不同的应用程序之间进行共享。所述系统具有主题和标记储存库、UI 储存库、可视化引擎、执行环境和 UI 服务。所述方法包括以下步骤：解析 XML 定义、应用主题和标记特征、向执行环境提供屏幕模型、对用户接口进行可视化以及事件处理。

根据本发明，提出了一种用于产生用于显示在无线设备的用户接口上的屏幕表示的方法，将屏幕表示定义为以结构定义语言来表达的用户接口定义集合，结构定义语言被配置为由被配置为在无线设备上运行的多个应用程序所引用，所述方法的特征在于包括以下步骤：接收来自多个应用程序中的第一应用程序的对屏幕表示的请求；从无线设备上的存储器中检索与屏幕表示相对应的用户接口定义集合；解析用户接口定义的结构定义语言，以确定屏幕表示的功能特征；将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；以及将屏幕模型提供给第一应用程序，以便以当前用户接口显示条件和当前屏幕值进行填充，以产生屏幕表示；其中将屏幕表示配置用于随后对用户接口的显示，以便通过用户事件与用户进行交互。

根据本发明的另一方面，提出了一种用于产生用于显示在无线设

备的用户接口上的屏幕表示的系统，将屏幕表示定义为以结构定义语言来表达的用户接口定义集合，结构定义语言被配置为由被配置为在无线设备上运行的多个应用程序所引用，所述系统的特征在于包括：存储器，用于存储由多个应用程序进行引用的多个用户接口定义集合；可视化引擎，用于接受多个应用程序中的第一应用程序的屏幕表示请求，以及用于解析从存储器中检索出的选定用户接口定义集合的结构定义语言，以确定屏幕表示的功能特征，选定的用户接口定义对应于所请求的屏幕表示；与可视化引擎相连的屏幕模块，用于将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；执行环境，用于向第一应用程序提供屏幕模型，以便以当前用户接口显示条件和当前屏幕值进行填充；以及用户接口服务，用于再现屏幕模型，以便向用户接口提供屏幕表示；其中无线设备的用户与用户接口上的屏幕表示进行交互。

根据本发明的另一方面，提出了一种计算机程序产品，用于产生用于显示在设备的用户接口（UI）上的屏幕表示，将屏幕表示定义为以配置用于由当在设备上提供时的多个应用程序所引用的结构定义语言来表达的 UI 定义集合，所述计算机程序产品包括：计算机可读介质；存储在所述计算机可读介质上的存储器模块，用于存储由多个应用程序进行引用的多个 UI 定义集合；存储在所述计算机可读介质上的可视化引擎，用于接受多个应用程序中的第一应用程序的屏幕表示请求，以及用于解析从存储器中检索出的选定 UI 定义集合的结构定义语言，以确定屏幕表示的功能特征，选定的 UI 定义对应于所请求的屏幕表示；与可视化引擎模块相连的屏幕模块，用于将外观特征应用于功能特征，以产生定义了屏幕表示的屏幕模型；以及存储在所述计算机可读介质上的再现模块，用于再现屏幕模型，以便向用户接口提供屏幕表示；其中设备的用户与用户接口上的屏幕表示进行交互。

附图说明

通过以下仅作为实例而参照附图做出的详细描述，这些和其他特征将变得更加清楚，其中：

图 1 是网络系统的方框图；

图 2 是图 1 所示的普通设备的方框图；

图 3 示出了与图 2 所示的设备的 UI 储存库交互的多个应用程序；
图 4 是使 UI 定义在图 2 所示的用户接口上可视化的系统；
图 5 示出了图 4 所示的 UI 存储库的 UI 定义层次；以及
图 6 是图 4 所示的系统的示例操作的流程图。

具体实施方式

网络系统

参照图 1，网络系统 10 包括多个普通终端设备 100，用于通过如但不局限于因特网的相连广域网(WAN)104 与一个或多个网络服务 106 交互。这些普通终端设备 100 可以是但并不局限于个人计算机 116、无线设备 101、PDA、自助信息站等。由服务 106 提供的普通服务可以是其他服务，如但并不局限于 SQL 数据库、基于 IDL 的 CORBA 和 RMI/IIOP 系统、遗留数据库、J2EE、SAP RFC 和 COM/DCOM 组件。此外，系统 10 还可以具有网关服务器 112，用于通过局域网 (LAN) 14 将桌面终端 116 与服务 106 相连。此外，系统 10 还可以具有无线网络 102，用于将无线设备 101 与 WAN 104 相连。应当意识到，其他设备和计算机（未示出）可以通过 WAN 104 和除图 1 所示的其他关联网络与网络服务 106 相连。此后，为了简单，将普通终端设备 100、无线设备 102 和个人计算机称为设备 100。为了简单，针对系统 10 的以下描述选择网络服务 106。但是，应当意识到，如果需要，可以用其他服务替换网络服务 106。此外，为了简单，此后将系统 10 的网络 102、104、112 称为网络 104。

再次参照图 1，设备 100 在与网络服务 106 进行通信时，分别发送和接收请求/相应消息 105。设备 100 可以通过使用消息报头信息和关联数据内容形式的的请求/响应消息 105，作为网络服务 106 的网络客户端进行操作，例如，从在线贸易商那里请求和接收产品价格和可用性。网络服务 106 是通信设备 100 上的客户端应用程序 302（参见图 2）通过无线网络 104 与之交互、以便向通信设备 100 的用户提供实用性的系统的示例。

再次参照图 1，为了满足适当的请求/相应消息 105，网络服务 106

可以通过多种协议（如但并不局限于 HTTP 和组件 API）与应用程序服务器 110 进行通信，以便在设备 100 上提供其时，向客户端应用程序 302（参见图 2）展示相关的商业逻辑（方法）。应用程序服务器 110 还可以包含网络服务 106 软件，从而网络服务 106 可以被认为是应用程序服务器 110 的子集。设备 100 的应用程序 302 可以使用应用程序服务器 110 的商业逻辑，类似于调用与对象有关的方法（或函数）。应当意识到，可以通过消息 105，经由网络 104，将客户端应用程序 302 与应用程序服务器 110 相关地直接下载/上载到设备 100。还应当意识到，设备 100 可以通过网络 104 与一个或多个网络服务 106 和关联应用程序服务器 110 进行通信。

服务器环境

参照图 1，网络服务 106 提供由设备 100 上的客户端应用程序 302（参见图 2）使用的信息消息 105。可选地，或此外，网络服务 106 可以接收和使用由设备 100 上所执行的客户端应用程序 302 所提供的信息消息 105，或者以设备 100 上所执行的客户端应用程序 302 的名义执行任务。可以将网络服务 106 定义为软件服务，其可以实现如利用登记在网络服务登记中的通用发现描述和集成（UDDI）中的网络服务描述语言（WSDL）所表达的接口，并能够通过如简单对象存取协议（SOAP）等适当的协议暴露在网络 104 上，通过消息 105 与客户端设备 100 进行通信。可选地，网络服务 106 可以使用其他已知的通信协议、消息 105 格式，以及可以按照除上述之外的其他网络服务语言来表达接口。

客户端环境

参照图 2，通过网络 104 传输组件应用程序 302，并加载到设备 100 的设备基础设施 204 的存储模块 210。可选地，可以通过串行连接、USB 连接、或如 IR、802.11 (x)、蓝牙 TM 等短距离无线通信系统（未示出）来加载组件应用程序 302。一旦被加载到设备 100 中，则可以由设备 100 上的执行环境 312 执行组件应用程序 302，如果需要，其

可以将组件应用程序 302 转换为本地代码，并通过设备基础设施 204 中的处理器 208。可选地，可以通过设备 100 上的其他软件模块（未示出）或操作系统来解释应用程序 302。在任何情况下，在由设备 100 提供的执行环境 312 下运行组件应用程序 302。执行环境 312 可以由智能软件框架 206 提供，所述智能软件框架 206 还能够提供用于管理和执行典型应用程序 302 行为（如持续性、消息发送、屏幕导航和表示等）的基本服务集合。

参照图 3，应用程序 302 可以是但并不局限于浏览器应用程序 302a、本地语言应用程序 302b 和/或基于容器的脚本/结构定义语言（如 XML）应用程序 302c，在适当的执行环境 312 下执行。设备 100 上所提供的每个应用程序 302a、b、c 可以使用用户接口（UI）储存库 310，从而使 UI 储存库 310 包含以结构定义语言（如但并不局限于 XML）描述的 UI 定义 600（参见图 5）。每个应用程序 302a、b、c 在 UI 储存库 310 中具有其自己的条目，其中存储了针对此应用程序 302a、b、c 的 UI 定义 600。UI 定义 600 由应用程序 302a、b、c 使用，以提供对用户接口 202 的输出，以便与设备 100 的用户进行交互。浏览器应用程序 302a 可以是在浏览器执行环境 312 下在设备 100 上执行的应用程序 302。浏览器应用程序 302a 的特征在于在设备 100 上较小的覆盖区（footprint），由于大多数应用程序逻辑均位于应用程序服务器上（即网络服务 106——参见图 1）。浏览器环境 312 提供了“沙箱”安全环境来执行浏览器应用程序 302a，并从而能够确保适当的访问控制。本地语言应用程序 302b 是以特定的语言实现的应用程序 302，所述语言是设备 100 的本地环境 312 所固有的——例如，C++、Java 等。本地应用程序 302b 可以使用设备 100 特征的扩展集合，但其很少可以在不同的设备 100 环境 312（例如，平台）之间移动。基于容器的脚本/XML 应用程序 302c 是使用脚本语言和以 XML 或其他结构定义语言所定义的元数据而定义的应用程序 302。可以在基于容器的运行时刻环境 312 下执行这些应用程序 302c。为了简单，此后将应用程序 302abc 称为应用程序 302。

再次参照图 1，可以对由设备 100 提供的客户端运行时刻环境进

行配置，从而使设备 100 作为网络服务 106 的网络客户端进行操作。应当意识到，客户端运行时刻环境也可以使设备 100 成为网络 104 上的任何其他普通服务的客户端，如但并不局限于普通计划定义服务。此外，框架 206 的特定功能可以包括但并不局限于对语言的支持、协调存储器分配、联网、I/O 操作期间的数据管理、协调设备 100 的输出设备上的图形以及提供对面向核心对象的类的存取以及支持文件/库。由设备 100 实现的运行时刻环境的示例可以包括如但并不局限于微软提出的公共语言运行时间（CLR）和 Sun 微系统公司提出的 Java 运行时刻环境（JRE）。

通信设备

再次参考图2，设备100是诸如但并不局限于移动电话、PDA、双向寻呼机或双模式通信设备的设备。所述设备100包括网络连接接口200，例如无线收发机或有线网络接口卡或调制解调器，通过连接218与设备基础设施204相连。在设备100的操作期间，连接接口200可通过无线链路（例如RF、IR等）与诸如无线网络102等网络104相连，这使设备100能够彼此通信且通过网络104与外部系统（诸如网络服务106）进行通信，并且协调客户端应用程序302和服务106之间的请求/响应消息105（见图1）。网络104支持设备和外部系统之间的请求/响应消息105的数据传输，所述外部系统与网络104相连。网络104还可以支持设备100和网络104外部的设备之间的电话呼叫的语音通信。无线网络102可以使用无线数据传输协议，例如但并不局限于DataTAC、GPRS或CDMA。

再次参考图2，设备100还具有通过连接222与设备基础设施204相连的用户接口202以便与用户（未示出）进行交互。用户接口202包括一个或多个用户输入设备，诸如但并不局限于QWERTY键盘、小键盘、跟踪轮、铁笔、鼠标、麦克风和用户输出设备（例如LCD屏幕显示器）和/或扬声器。如果屏幕是对触摸敏感的，则还可以将所述显示器用作由设备基础设施204控制的用户输入设备。由设备100的用户采用用户接口202来协调系统10（见图1）上的请求/响应消息105，如由框架206的客户端应用程序302所采用的那样。

再次参考图2，由设备基础设施204启动设备100的操作。所述设备基础设施204包括计算机处理器208和相关存储模块210。计算机处理器208通过执行由操作系统和位于存储模块210中的客户端应用程序302提供的相关指令，控制通信设备100的网络接口200、用户接口202和框架206的操作。所述存储模块还可以包括UI储存库310和主题和标记储存库410，如以下将详细描述的。应该意识到，设备基础设施204可以与处理器208相连的计算机可读存储介质212，用于向处理器提供指令和/或加载/更新存储模块210的客户端应用程序302。所述计算机可读存储介质212可以包括硬件和/或软件，仅作为示例，例如磁盘、磁带、可光学读取的介质（例如CD/DVD ROM）和存储卡。在每一种情况下，计算机可读介质212可以采用小型盘、软盘、盒式磁带、硬盘驱动器、固态存储卡或RAM的形式，设置在存储模块210中。应该注意，以上所列出的计算机可读介质212可以或者单独使用或者组合使用。

设备的框架

再次参考图2，设备100的框架206通过连接220与设备基础设施204相连。设备100的框架206具有优选地能够产生、主持和执行客户端应用程序302的执行环境312。框架206可以被认为是提供基本服务集合304以管理和执行典型应用程序302行为（例如但并不局限于持续性、提供、消息传递、屏幕导航和用户接口/屏幕服务）的智能软件框架206。因此，框架206为客户应用程序302提供适当的执行环境，并且是与处理器208的设备100功能的接口，并且是设备基础设施204的相关操作系统。框架206通过优选地在设备100上提供受控、安全和稳定的环境来提供执行环境312，其中应用程序302在所述环境下执行。

再次参考图2，在特定的服务并未被包括作为客户端应用程序302的一部分或未被接收为作为应用程序302的一部分的单独组件（未示出）的情况下，框架206可以向客户端应用程序302提供服务（标准普通服务集合）。如果需要，应用程序302具有与服务304之间的通信。应当意识到，设备基础设施204（参见图2）的操作系统的一部分可以表示任何服务304。应当意识到，通信设备100的服务304可以向应用程序

302提供功能性，其可以包括上述服务。此外，可以将服务304与应用程序302进行集成，而不是作为单独的框架来提供。在任何情况下，组件应用程序302可以通过集成的和/或单独的服务304来使用通信设备100的功能，如以下所讨论的那样。服务304包括当在用户接口202的输出设备上输出其时管理应用程序302的表示的UI服务308（参见图4），如可视化引擎306（参见图4）所提供的那样。服务304的提供服务可以管理通信设备100上的软件应用程序302的提供。服务304的持续服务可以允许应用程序302将数据存储在存储模块210中，以及访问UI储存库310和主题/标记储存库410。

用于产生UI屏幕表示的UI系统

参照图4，用于对UI定义进行可视化的系统300包括5个基本模块，即：

- 主题和标记储存库410；
- UI储存库310；
- 可视化引擎306；
- 执行环境312；以及
- UI服务308。

可以将UI服务308定义为负责再现用户接口202的UI控制并解释来自那里的用户输入的服务304。UI服务308典型地专用于不同的设备100平台（即本地）。可以将执行环境312定义为执行所有相应应用程序302的环境。在一些实现中，其可以是Java虚拟机、基于组件的框架或简单地用于运行设备的本地应用程序的环境。可视化引擎306可以定义为用于根据在环境312下所执行的应用程序302的请求，解析存储在UI储存库310中的UI XML定义600并解释其的引擎。UI定义600提供显示在用户接口202上的屏幕元素的功能特征。可视化引擎306构建针对用户接口202的UI屏幕表示602（参见图5）的本地屏幕模型307，然后，UI服务可以代表所涉及的应用程序302呈现给用户。UI储存库310可以定义为包括设备100上的所有应用程序302的UI定义600（参见图5）的储存库。主题和标记储存库410可以定义为再现UI定义600的信息和规则

的储存库，专用于优选地由设备100的用户所指定（至少部分指定）的当前主题和优选地由设备100的通信公司所选择的标记。主题的示例包括如自然和技术喜好等背景主题。标记示例包括颜色、布置和标志图细节。来自主题和标记储存库410的信息和再现规则影响到可视化引擎306如何从UI储存库310中，针对选定UI定义600，通过屏幕模型307，来产生UI屏幕表示602。储存库410的再现信息和规则提供了显示在用户接口202上的屏幕元素的外观特征。

UI定义600

以XML或任何其他结构定义语言来定义储存库310中的UI定义600，并由可视化引擎306在应用程序302的提供阶段和/或执行阶段期间进行解析（参见图4）。定义600提供显示在用户接口202上的屏幕元素的功能特征，并可以包括以下项目，例如但并不局限于：屏幕布局、屏幕内的控制、控制布局、事件处理和多种可视化属性。参照图5，定义600包括UI屏幕表示602，可以将其定义为定义了用户接口202（参见图2）的UI元素集合，在给定的时刻向用户展示。UI屏幕表示602可以具有不同的属性，例如但并不局限于：逻辑名称；标题；全屏或对角线模式；前景和背景色；以及缺省字体。定义600还可以具有事件处理定义604，可以将其定义为规定了当UI屏幕表示602在用户接口202上有效时，应用程序302应当如何处理来自用户的事件的屏幕元素。定义604包括将应用程序302插入处理中的事件的列表。这些事件可以触发要发送给应用程序302的消息处理器（例如）的消息，或调用具有特定命名约定的方法。对于基于容器的脚本/XML应用程序302，事件处理定义可以规定要执行或导航到接口202的另一屏幕的脚本块。定义600还包括屏幕菜单606，可以将其定义为规定了当屏幕表示602在用户接口202上有效时，可访问的菜单项集合的屏幕元素。在菜单中对菜单项进行排列，并具有关联动作。菜单项动作是由事件处理定义使用的UI事件。定义600还包括UI布局608，其定义了UI控制610在屏幕表示602上的订单和定位。UI布局608影响其所包含的UI控制610。定义600还包括能够定义为用于构建屏幕显示602的用户接口元素的UI控制610。公共UI控

制610是但并不局限于：编辑框、按钮、选项控制、图像控制、滚动条和静态文本。

在应用程序之间共享UI定义

参照图4和5，可以在执行环境312的不同应用程序302之间共享UI定义600。这意味着一个应用程序302可以根据存储在另一应用程序302的UI储存库310条目中的UI定义600来实例化屏幕表示602。这有助于节约开发劳动，以实现应用程序302之间的一致“感观”，并提供更为容易的维护。

一个应用程序302可以通过引用以拥有定义600的应用程序302的应用程序302名称为前缀的UI定义600，来实例化来自属于另一应用程序302的UI定义600的屏幕表示602。例如，可以使用单个的斜线来作为应用程序302的名称与由定义600所产生的引用屏幕表示602的名称之间的分界符。例如，如果应用程序“A”需要引用应用程序“B”的UI储存库310条目中所定义的屏幕表示602“OrderStatus”，则应当在应用代码中通过“B\OrderStatus”来查阅屏幕表示602，以便链接到定义600，产生“OrderStatus”屏幕表示602。按照这种方式，不同的应用程序302可以共享和执行UI定义600。应当意识到，有效的应用程序302可以负责处理针对由共享UI定义600所构建的屏幕表示602的任何用户事件。在上述示例中，应用程序“A”还可以提供如应用程序“B”所实现的“OrderStatus”屏幕表示602所需的事件处理。

可视化引擎306可以支持和实现能够由设备100上的所有应用程序302重新使用的预定义全局UI定义600。通用全局UI定义600的示例是但并不局限于：

1、 1) 对话：

- URL条目对话
- 登录对话
- 确认对话
- 搜索对话
- 等等；

-
- 2、 2) 样式、主题；以及
 - 3、 3) 公共布局、控制、动画等。

根据无线设备目标的区域，频繁使用的UI定义600的集合可能会发生波动。例如，对于邮件中心设备100，用于写作新电子邮件的形式将是频繁使用的UI定义600，因而适合于将其包括在UI定义600的全局集合中。

平台无关

参照图4，系统300是平台无关的，由于可以按照平台无关的方式定义应用程序的用户接口202。可视化引擎306是负责根据每个UI定义600来构建平台相关屏幕的模块。为了在不同的平台上重新使用UI定义600，可以针对目标平台专门提供可视化引擎306。应当意识到，如果需要，可以调整可视化引擎306以适应于两个或更多个设备100平台。

主题和标记

参照图4和5，系统300和相关方法可以允许设备100上的所有应用程序302的无缝标记（branding）。设备100，例如无线的，通常是针对特定的提供商而进行标记的对象，所述提供商或者为无线服务的无线通信公司或其他提供商。通过标记设备100，无线提供商可以将所提供的特征集合与用户接口202的提供商专用“感观”相关联。与竞争对手所具有的产品相比，通过标记其产品，提供商还可以试图创建更容易吸引用户的用户接口202。系统300和相关方法从UI定义600分离出储存库410中的标记信息。可以根据应用程序开发，单独创建标记信息，以及可以针对不同的提供商进行定制。由于在可视化引擎306级考虑标记信息，应用特定的标记简档表可以影响无线设备100上的所有应用程序302。额外安装的任何应用程序302也将考虑设备100上的标记信息。

设备100的另一特征在于使用户能够根据特定的个人喜好来定制用户接口202的“感观”的能力。由于以下事实而特别关注此特征：与桌面计算机100不同，无线设备100可能被认为是更加私人化。无线设备100可以由用户携带，并很少在几个用户之间共享。使用相同的解决

方案作为标记，系统300和相关方法通过支持UI主题，提供了用于定制安装在设备100上的所有应用程序302的用户接口202的机制。可以将所述主题定义为定制设置的集合。可以将多个主题存储在储存库410中，并根据用户的请求加以应用。储存库410的规则和信息提供了显示在用户接口202上的屏幕元素的外观特征，如但并不局限于针对如一天的不同时间、一周的不同日子或用户的不同心情和视觉偏好而定制的主题和布局。

用于处理UI XML储存库的操作

下面，参照图4、5和6，对操作700进行描述。

步骤701：解析XML定义

在应用程序302启动时，可视化引擎306从UI储存库310中检索应用程序的UI定义600。当进行屏幕激活请求时，可视化引擎306找出屏幕的XML UI定义600，并对其进行解析。如果对UI定义的引用属于另一应用程序302，可视化引擎306从UI储存库310中检索所请求的定义600。对于UI定义600中的每一项，创建相应的平台专用UI元素，并将其添加为屏幕的本地模型307。例如，当遇到编辑控制的定义600时，实例化编辑框的平台专用类，并添加为屏幕的模型307。本地屏幕模型307是平台专用的，并在屏幕上提供UI定义600的有效再现。可以将额外的UI元素添加为模型307，以便改善用户在特定平台上的体验。应当意识到，也可以将屏幕模型307产生为平台无关模型，然后根据需要，转换到设备100平台。

步骤702：应用主题和标记特性

在构建屏幕模型307期间，可视化引擎306使用主题和标记储存库410中可用的信息/规则集合，以便将定制的“感观”赋予UI元素。主题和标记储存库410包含针对需要定制外观的所有UI元素的再现信息。

步骤703：为执行环境312提供屏幕模型307

一旦已经构建了屏幕模型307，可视化引擎306就将其传递给执行

环境312。通过执行环境312，使屏幕模型307可用于针对额外定制的请求应用程序302，如果可应用的话，并产生针对用户接口202的动态屏幕表示602。应用程序302与屏幕表示602的这种交互作用可以包括表示用户接口202上的当前显示条件的当前值的填充。由于应用程序302可以自由地操纵屏幕模型307，系统300和操作700可以允许构建丰富且动态的屏幕表示602。还应当意识到，可视化引擎可以负责以当前屏幕值全部或部分填充屏幕表示602。

步骤704：对用户接口进行可视化

在此阶段，应用程序302向UI服务308提交屏幕模型307。UI服务308再现模307中的UI元素，并登记应用程序302，以便任何事件处理。

步骤705：事件处理

UI服务308将接口202上的任何用户事件传播回应用程序302，作为对应用程序逻辑的输入。应用程序302应当对事件进行处理，并将控制返回给UI服务308。对事件进行处理可以涉及对新屏幕的导航或向用户发送可视反馈。此处理可以涉及从UI储存库310中检索新UI定义600，以及创建适当的新屏幕模型307，如上所述，或者可以简单地涉及通过UI服务308，对用户接口上的当前屏幕表示602的控制的更新。

样本UI定义

以下是向用户提示用户名和密码的屏幕表示602的样本UI定义600。在UI定义600中定义了两个导航按钮610——btnRegister和btnLogin。这些按钮610可以分别导航到新用户注册屏幕或尝试登录所输入的用户。

```
<xmlScreen name="scrLogin" title="Login" dialog="true"
bgImage="backg.jpg">
<xmlLayout type="vertical">
<xmlLabel name="lblUserName" value="User Name: "/>
```

```

<xmlEdit name="edUserName" type="char"/>
<xmlLabel name="lblPassword" value="Password: "/>
<xmlEdit name="edPassword" type="pwd"/> 5
<xmlButton name="btnLogin" value="Login">
    <event type="onClick" handler="hLogin"/>
</xmlButton>
<xmlButton name="btnRegister" value="Register">
    <event type="onClick" screen="scrRegisterUser"/>
</xmlButton>
</xmlLayout>
</xmlScreen>

```

以下是对上述屏幕表示602的解释：

- <xmlScreen - 定义UI屏幕
 - o name="scrLogin" - 定义屏幕的逻辑名称。之后，可以通过其逻辑名称引用所述屏幕
 - o title="Login" - 定义屏幕的标题
 - o dialog="true" - 将屏幕定义为与全屏相对的对话
 - o bgImage="backg" - 定义屏幕的背景图像
 - <xmlLayout type="vertical"> - 定义UI控制610的垂直次序
 - <xmlLabel name="lblUserName" value="User Name: "/> - 以逻辑名称“lblUserName”和数值“User Name:”定义屏幕上的静态标签
 - <xmlEdit name="edUserName" - 以逻辑名称“edUserName”定义编辑框
 - o type="char" - 规定编辑框应当接受任何字符和数字
 - <xmlButton name="btnLogin" value="Login"> - 以逻辑名称“btnLogin”和标签“Login”定义按钮
 - o <event type="onClick" handler="hLogin"/> - 定义在点击按钮时应当用于处理用户事件的句柄。“hLogin”为事件句柄的名称

o <event type="onClick" screen="scrRegisterUser"/> - 以逻辑名称“scrRegisterUser”定义到另一UI定义600的转换

尽管这里已经对一个或多个典型系统和方法进行了公开，多种变体对本领域的普通技术人员而言是显而易见的，而且这种变体在本申请的范围之内。例如，尽管在所提供的示例中使用 XML，也可以使用其他语言和语言变体来定义应用程序 302。

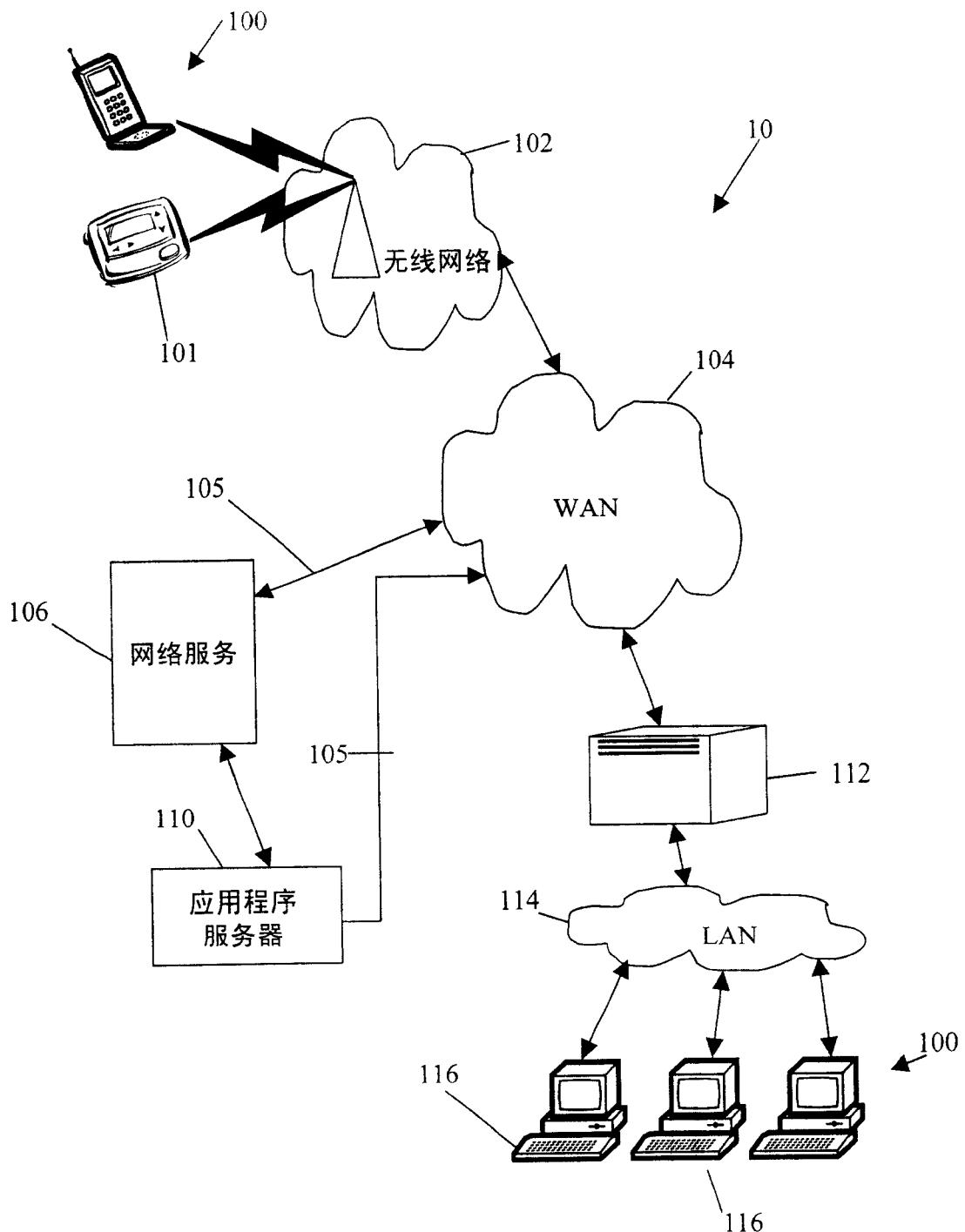


图 1

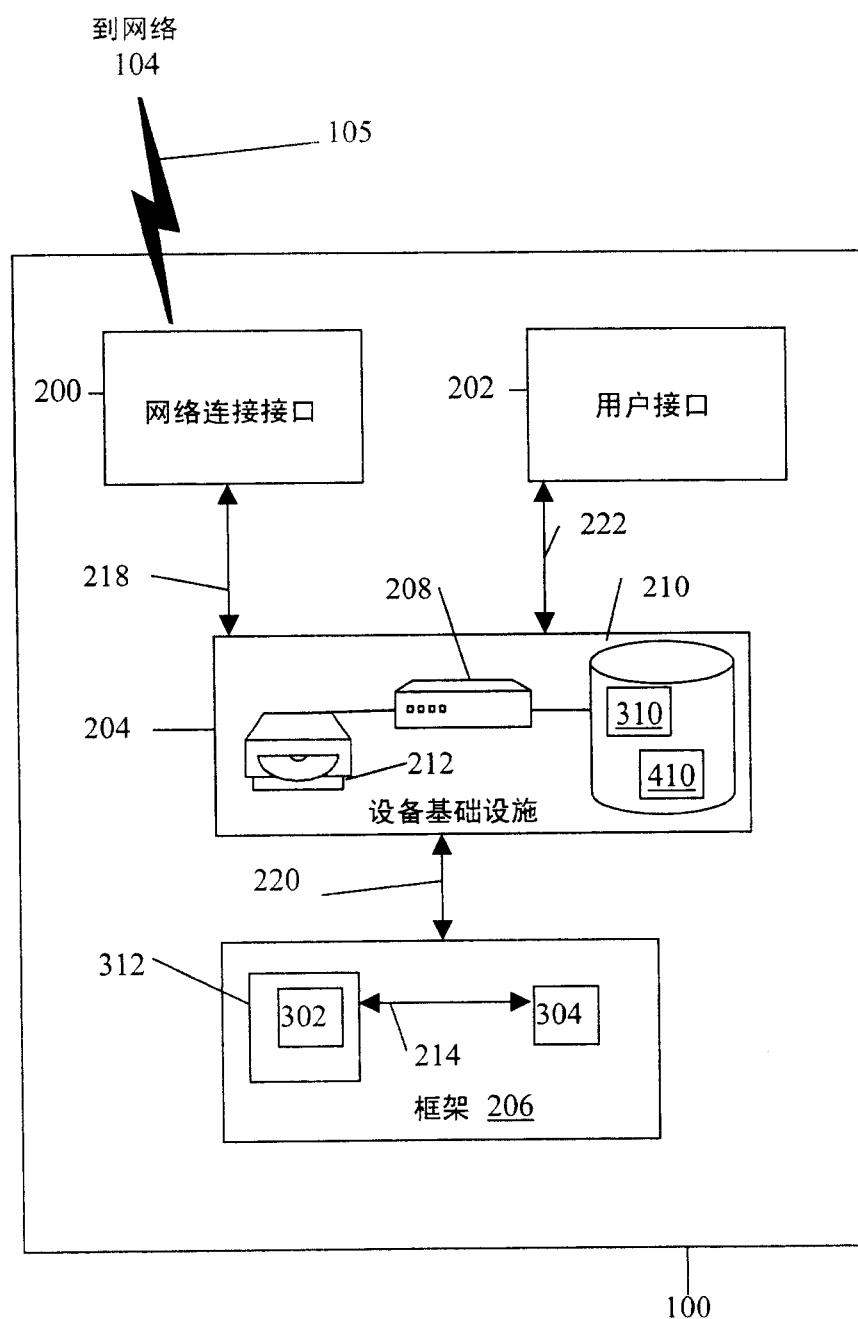


图 2

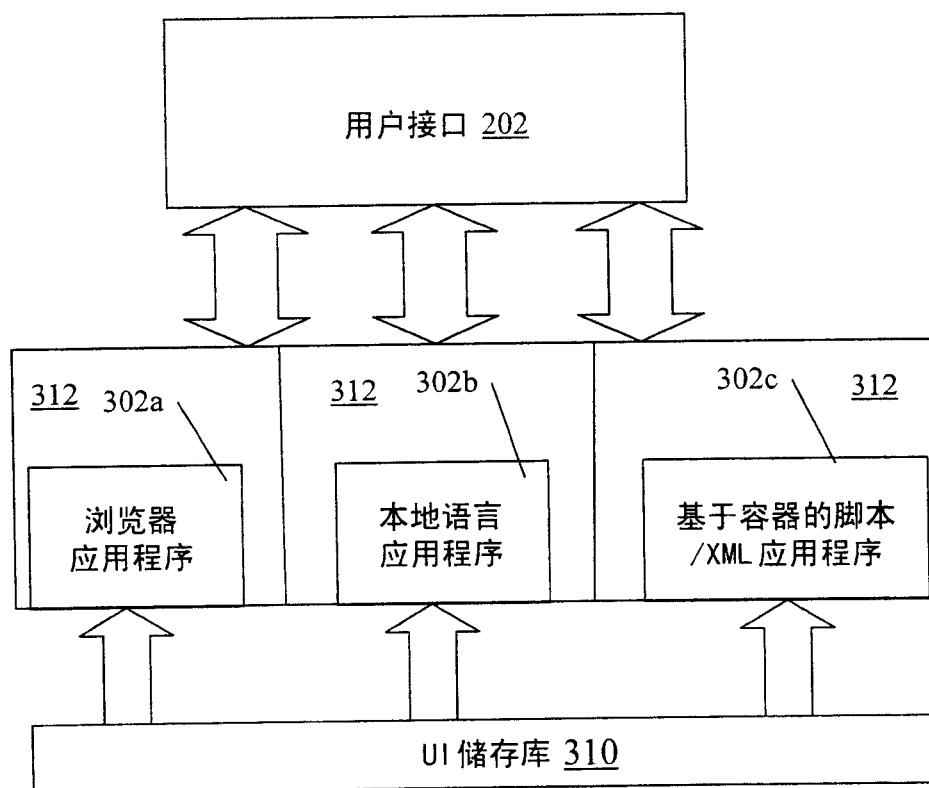
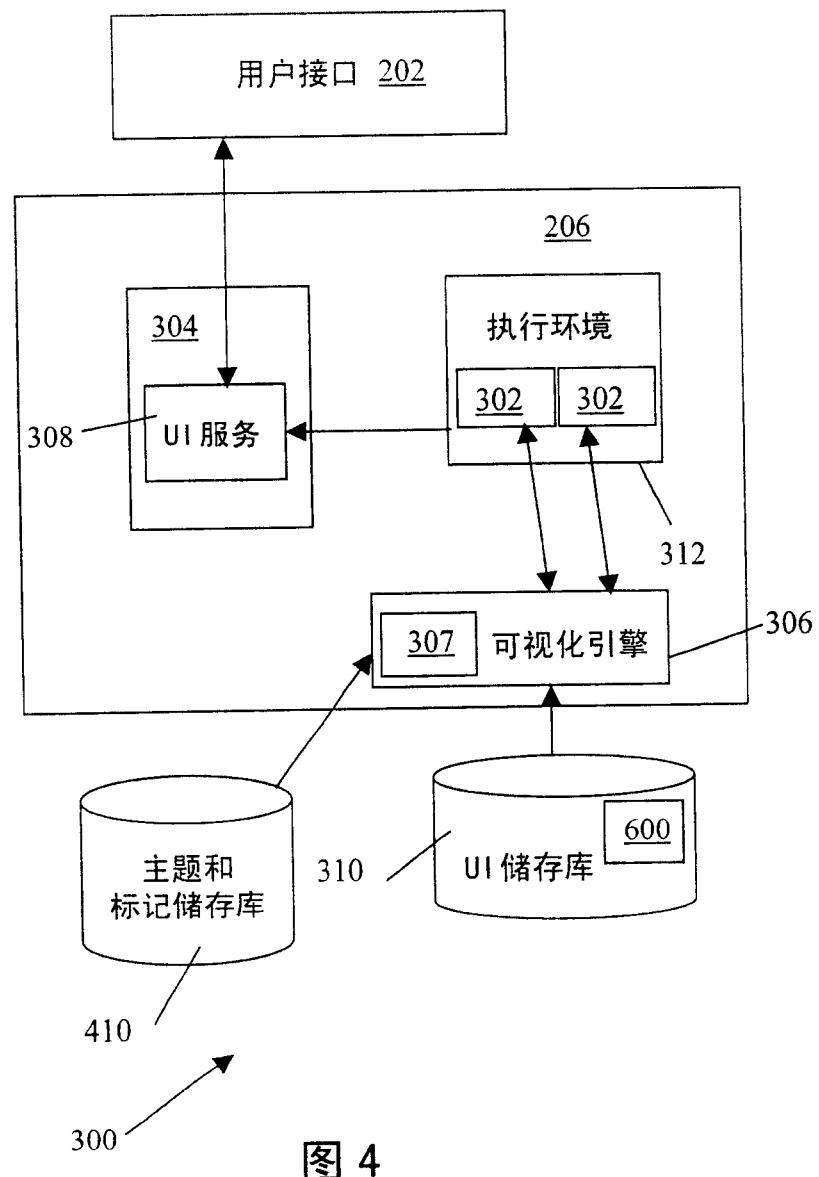


图 3



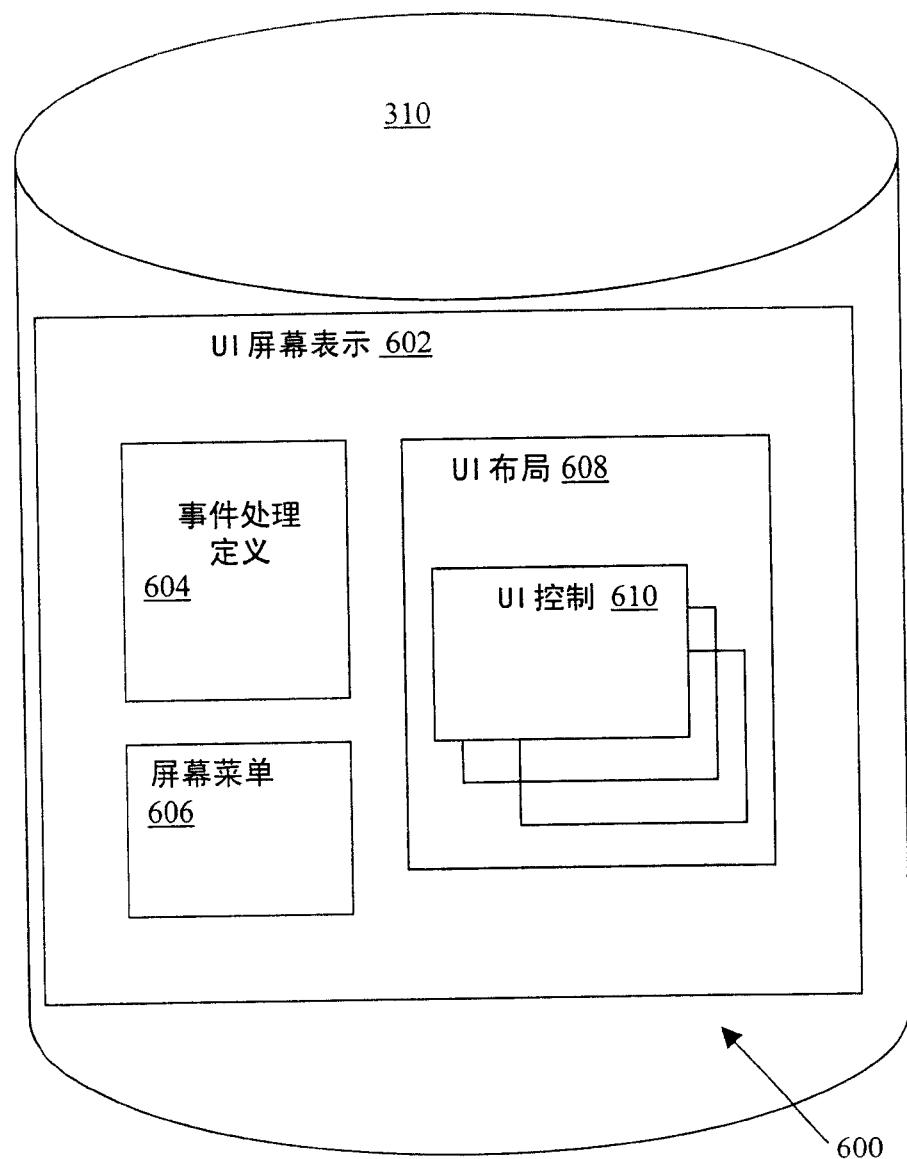


图 5

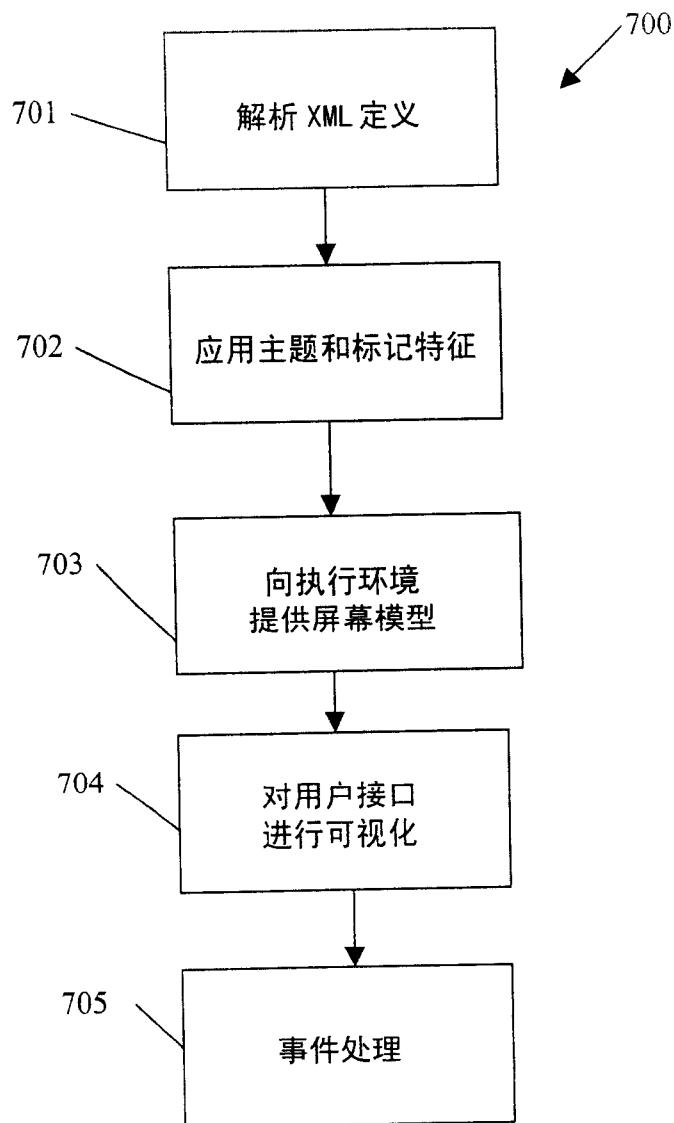


图 6