



(12) 发明专利

(10) 授权公告号 CN 1470984 B

(45) 授权公告日 2012.12.12

(21) 申请号 03145725.8

US 5778227 A, 1998.07.07,

(22) 申请日 2003.06.30

审查员 邹斌

(30) 优先权数据

10/187,012 2002.06.28 US

(73) 专利权人 微软公司

地址 美国华盛顿州

(72) 发明人 J·L·波格丹 M·J·菲诺齐奥

N·M·克雷默

(74) 专利代理机构 上海专利商标事务所有限公司 31100

代理人 陆嘉

(51) Int. Cl.

G06F 9/40 (2006.01)

G06F 9/45 (2006.01)

(56) 对比文件

US 5732271 A, 1998.03.24,

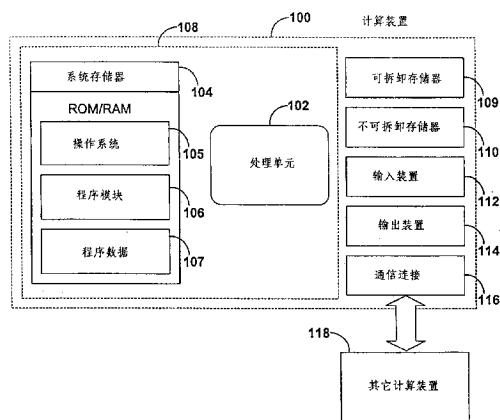
权利要求书 1 页 说明书 9 页 附图 6 页

(54) 发明名称

使特性与对象相关联的系统和方法

(57) 摘要

所描述的是一种机制，用于把对象的新功能表示为特性，没有把所述特性构造到派生对象的类别中。尤其，所述机制使一种类别中的特性与另一个类别相关联。一种计算机可读出媒体，它包括对象，所述对象具有在第一特性组中的特性，进一步包括数据结构。数据结构包括对于第二特性组中的每一个的一些定义，并包括至少一种静态方法。静态方法与第二特性组中的一个特性相关联，并包括第一参数。第一参数唯一地定义一个特性。可操作静态方法使一个特性与对象相关联而无需指定对于对象中的一个特性的明确参考。在运行时间期间注册特性以便接收唯一的识别符。



1. 一种由计算机实现的装置,其特征在于,包括:

实现包括多个特性的基本类的装置;以及

实现包括多个附加特性的附加类的装置,附加特性中的每一个能够与从基本类派生的一个对象的一个实例相关联,所述附加类进一步包括至少两种静态方法,用于使多个附加特性中的一个附加特性与对象的实例相关联,所述静态方法支持强类型的语法,强类型允许所述静态方法在执行前进行检测。

2. 如权利要求1所述的装置,其特征在于,从基本类派生的对象继承来自节点类的至少一种方法,当一个附加特性与对象的实例相关联时,所述一种方法由所述静态方法所调用。

3. 如权利要求2所述的装置,其特征在于,所述静态方法和一种方法中的每一个方法包括用于传递一个附加特性的唯一识别符的第一参数,所述一个附加特性与对象的实例相关联。

4. 如权利要求3所述的装置,其特征在于,在运行时刻注册一个附加特性之后可得到唯一识别符。

5. 如权利要求2所述的装置,其特征在于,一种方法支持松散类型的语法。

6. 如权利要求1所述的装置,其特征在于,相关联一个附加特性包括从对象的父对象检索所述对象的值。

7. 如权利要求1所述的装置,其特征在于,相关联一个附加特性包括来自特性页检索对象的值。

8. 一种使用对象的计算机系统,其特征在于,包括:

实现从第一类派生的第一对象的装置,所述第一对象包括第一组特性;

实现把第二组特性提供给所述第一对象的第二类的装置,所述第二类包括至少两种静态方法,所述静态方法用来根据与所述第一对象相关联的操作把第二组特性中之一附加到第一对象,所述静态方法支持强类型的语法,强类型允许所述静态方法在执行前进行检测。

9. 如权利要求8所述的计算机系统,其特征在于,所述静态方法包括第一参数,用于唯一地识别所述操作请求第二组特性中的哪一个特性。

10. 如权利要求9所述的计算机系统,其特征在于,在第一参数可以唯一地识别一个特性之前该特性被注册。

使特性与对象相关联的系统和方法

(1) 技术领域

[0001] 本发明一般涉及软件应用程序,尤其,涉及管理在软件应用程序中的对象特性的机制。

(2) 背景技术

[0002] 当今大多数编程模型都支持类别的概念。一般在具有分支的分层树中构成这些类别,所述分支表示在类别层次中的不同类别。当两个分支处于不同级别时,较低的分支表示子类。子类继承从与上分支(例如,父类)相关联的类别来的信息。当两个分支处于相同级别时,把类别称为姐妹类。为了下述讨论的目的,可以分别使用术语下级和上级来指出子类和父类。在层次中最上面的分支表示在分层类别树中的基本类。一般,在每个类别中的信息包括特性、方法和事件。特性描述与类别相关联的特征。例如,按钮类可以具有诸如宽度、背景颜色、字体类型、可见性和按下之类的特性。当建立这些类别中的一个实例时,就创建该类别的对象。在对象中的每个特性都有相关联的值,在运行时间操作期间,可以查询和设置该值。如果语法有较强的代表性,则可以期望这个查询和设置的值符合特定的数据类型。希望语法具有较强的代表性,因为可以在运行时间操作之前检测软件应用程序中的差错。

[0003] 一旦类别层次就位,把新功能添加到类别层次的对象中就会出现问题。在一种编程模型中,把新功能强加到基本类中。当这样做时,基本类变成极大(例如,一百种方法、五十个特性以及二十件事件),这导致对象层次几乎不可管理。这种编程模型的一个不希望有的结局是在开发者把所要求的特性实施在他们创建的对象中之前,充分了解在基本类中的特性、方法和事件的数目对于开发者来说变成比什么都重要。这种编程模型的再另一个不希望有的结局是由于这些特性值事实上是局部存储的,存储器的需求变成极大。因为局部地存储这些值,用这个编程模型创建的应用程序就不能较好地换算。

[0004] 这些编程模型还导致对于第三方开发者的其它问题。希望添加新功能的第三方开发者必须在层次的底部添加一个子类,层次中的其它类别不可得到新功能。因此,第三方开发者可能需要把新功能添加到数个子类。可以想象,这导致代码复制,它影响了对象层次的可保持性。为了上述原因,极不希望有这种编程模型。

[0005] 直到有了本发明,熟悉本技术领域的人员才懂得一种编程模型,该编程模型允许把新功能提供给现有类别层次中的类别而没有上述缺点。

(3) 发明内容

[0006] 本发明提供一种机制,该机制允许把新功能提供给类别而无需使新功能变成类别的永久部分。此外,本发明允许把新功能表示为不是构造到类别中的一个特性。一般,本发明提供一种机制,使一个类别中的特性与另一个类别相关联。这种关联是易于修改的,以致可以使其它特性组与所述类别相关联。

[0007] 在一个实施例中,计算机可读出媒体包括具有第一特性组中的特性的一个对象,

所述计算机可读出媒体进一步包括一种数据结构。数据结构包括第二特性组中的每一个的定义，并包括至少一个静态方法。静态方法与第二特性组当中的一个特性相关联，并包括第一参数。第一参数唯一地识别一个特性。可操作静态方法使一个特性与对象相关联而无需指定对象中的一个特性的明确的参考。

[0008] 在本发明的一个方面，静态方法支持一种打印成黑体的语法。

[0009] 在本发明的另一个方面，静态方法包括检索对象的一个值而无需把所述值局部地存储在对象上。可以从诸如父对象或特性表 (property sheet) 之类的几个级别检索所述值。

[0010] 本发明的一个优点是把特性分割成一个或多个子集允许特性的每个子集变得更容易保存。本发明的另外的优点是更能扩展对象的层次，并允许开发者把功能添加到影响基本类和所有下级的对象层次。

[0011] 本发明的另外的优点是管理对象中特性的存储变得更有效和方便。

[0012] 本发明再另外的优点是编程模型在有计划的或标记的环境中操作。此外，编程模型在打印成黑体的编程语言（诸如 C++ 和 C#）中操作。此外，编程模型支持特性表、变更通知以及值继承。

[0013] 本发明的另外的优点是可以同时存在独立的程序库，因为由于所附类别的名称有效地变成特性名称的一部分而名称冲突的可能性较小。因此，如果两个不同的开发者每人都创建名称为“颜色”的一个附加特性，则两个附加特性不会冲突。

(4) 附图说明

[0014] 图 1 示出可以在本发明的一个示例实施例中使用的示例计算装置；

[0015] 图 2 是可以与图 1 的计算装置一起创建的示例显示器；

[0016] 图 3 是根据本发明的编程模型的图形表示，所述编程模型允许特性从一种类别附加到另一种类别；

[0017] 图 4 示出用于执行图 3 中示出的编程模型的几种示例语法；

[0018] 图 5 是根据本发明的逻辑流程图，示出设置一个值的一个过程；

[0019] 图 6 是根据本发明的逻辑流程图，示出检索一个值的一个过程。

(5) 具体实施方式

[0020] 简单地说，本发明提供一种编程模型，所述编程模型允许把新功能提供给一个类别而无需使新功能变成类别的一部分。此外，本发明允许把新功能表示为不构造到类别中的一个特性。一般，本发明提供一种机制，用于使一种类别中的特性与另一种类别相关联。在阅读下面详细说明之后会明白，本发明的编程模型把动态特性提供给对象而无需把特性构造到对象中。

[0021] 参考图 1，实施本发明的一个示例系统包括诸如计算装置 100 之类的计算装置。在一种极基本的配置中，计算装置 100 一般包括至少一个处理单元 102 以及系统存储器 104。根据计算装置的确切的配置以及类型，系统存储器 104 可以是易失性的（诸如 RAM）、非易失性的（诸如 ROM、快闪存储器等）、或它们两者的某些组合。系统存储器 104 一般包括操作系统 105、一个或多个程序模块 106，并可以包括程序数据 107。程序模块 106 的例子包括

来自 Redmond 的微软公司的 Visual Studio IntelliSense、WA 以及利用对象程序库的其它软件编程环境。此外，程序模块 106 包括使用软件 - 编程环境创建的软件应用程序。当在处理单元 102 上执行这些软件应用程序时，特性引擎根据本发明的编程模型处理软件应用程序。特性引擎可能是操作系统 105 的一部分，或可能是另一个程序模块 106。通过在图 1 中虚线 108 中的这些元件示出计算装置 100 的这种基本配置。

[0022] 计算装置 100 可以具有另外的特征或功能。例如，计算装置 100 也可能包括另外的数据存储装置（可拆卸的和不可拆卸的），例如，诸如磁盘、光盘或磁带。在图 1 中由可拆卸的存储器 109 和不可拆卸的存储器 110 来示出这种另外的存储器。计算机存储媒体可以包括以任何用于存储信息（诸如计算机可读出指令、数据结构、程序模块、或其它数据）的方法或技术实施的易失性和非易失性、可拆卸和不可拆卸媒体。系统存储器 104、可拆卸的存储器 109 和不可拆卸的存储器 110 是所有计算机存储媒体的例子。计算机存储媒体包括，但是不限于，RAM、ROM、EEPROM、快闪存储器或其它存储器技术、CD-ROM、数字通用盘 (DVD) 或其它光存储器、磁盒、磁带、磁盘存储器或其它磁存储装置、或可以用来存储所需要的信息和可以通过计算装置 100 访问的任何其它媒体。任何如此的计算机存储媒体可以是计算装置 100 的一部分。计算装置 100 还具有诸如键盘、鼠标、笔、话音输入装置、触摸输入装置等输入装置 112。还可以包括诸如显示器、扬声器、打印机之类的输出装置 114。这些装置是本技术领域中众知的，这里不需要用大篇幅来描述。

[0023] 计算装置 100 还可以包括允许所述装置，诸如通过网络，与其它计算装置 118 进行通信的通信连接 116。通信连接 116 是通信媒体的一个例子。通信媒体一般可以通过计算机可读出指令、数据结构、程序模块、或在调制数据信号中的其它数据（诸如载波波形与其它传递机制）来实施，并包括任何信息传递媒体。术语“调制数据信号”是指一种信号，所述信号具有它的一个或多个特征组或按把信息编码在信号中这样的方式而改变。作为例子，而不是作为限制，通信媒体包括诸如有线网络与直接有线连接之类的有线媒体，以及诸如声音、射频、红外线以及其它无线媒体之类的无线媒体。

[0024] 图 2 是可以通过图 1 中编程环境中的应用程序创建的一种示例显示。当可以在各种环境中使用本发明的编程模型时，通过不作为限制的例子，图 2 示出在用户 - 界面环境中的编程模型的应用。因此，在这个例子中，图 1 中的应用程序中之一创建图 2 中示出的显示 200。显示 200 包括对话框 202。对话框 202 可以包括许多其它控制（即，对象）。在这个例子中，对话框 202 包括列表框 204、编辑框 206 以及具有两个按钮对象（例如，同意 (OK) 按钮 210 和取消 (CANCEL) 按钮 212）的容器对象 208。如下面将连同图 3 进行的更详细的讨论，使用传统的编程技术，这些对象（例如，202-212）中的每一个都是从基本对象（例如，元类）派生的子对象。此外，根据本发明的一个实施例，还可以从节点类派生每个子对象。如下所述，这个节点类提供附加特性的“附加”特征。

[0025] 虽然显示 200 的外观显现出与使用现有编程模型创建的显示很相似，但是创建显示 200 的机制是十分不同的。在现有的编程模型中，实施显示 200 的软件代码具有对象的性能和结合在每个对象本身中或父对象之一中的对象外观两种特性。然而，如下详细描述，根据本发明，分立的类别提供对象 202-212 的动作和外观而无需依靠两种类别之间的明确的参考。

[0026] 这允许开发者修改动作部分或外观部分而无需修改整个对象。因此，在这个用

户 - 界面例子中, 每个对象分解成两部分。一部分涉及与对象相关联的动作 (即, 性能), 而另一部分涉及对象的外观 (即, 绘制)。因为绘制是独立于对象的特性的, 所以可以容易地修改对象的外观来改变对象显现的方式。

[0027] 当所述用户 - 界面例子方便地把对象的特性分解成性能组和外观组时, 发明者已经确定可以把各种环境中许多对象的特性方便地分割成两个或多个相关的组。然后, 根据本发明, 这些相关组可以“隐含地”相互关联, 以充分地实现所述对象。与现有编程模型比较, 这种“隐含地”相关联提供几个优点。优点之一是第三方开发者可以方便地修改与他们的应用程序有关的相关组, 同时不碰其它组。这大大地简化了开发者为了改变对象的功能性而需要知道的信息量。此外, 对象层次变得更便于管理。另外的优点是减少了存储器, 因为每个对象不需要保存当前状态的局部值。而是, 将在下面连同图 6 一起详述, 当需要时可以从各种源检索状态。

[0028] 图 3 是编程模型 300 的图形表示, 所述编程模型根据本发明提供从一种类别到另一种类别“隐含地”与特性的相关组相关联的机制。相似于其它编程模型, 本发明支持基本类 302。基本类 302 包括与基本类 302 相关联的第一特性集 304。此外, 基本类 302 包括方法集 306 和事件集 308。然而, 根据本发明, 从节点类 301 派生基本类。节点类 301 提供 SetValue() 方法 307 (设置值 () 方法) 以及 GetValue() 方法 309 (得到值 () 方法)。

[0029] 第二特性集 324 包括在附加类 322 中。附加类 322 还包括与第二特性集 324 中的每一个相关联的至少两种静态方法 (例如, SetFont() 326 (设置字体) 以及 GetFont() 330 (得到字体))。可以把这些静态方法考虑为可以访问一些对象的全局方法, 所述一些对象已经注册与静态方法相关联的特性。提供附加类 322 的开发者负责写入这些静态方法 326 和 330 中的每一个的代码。代码包括到节点类 301 提供的方法中之一的调用。例如, 对于特定的识别符 PropertyID (特性识别符) 和给定的字体值, 调用 SetFont() 326 的应用程序最终将执行到节点类 301 中的 SetValue() 307 的调用。SetFont() 326 传递到 SetValue() 307 的调用的参数将包括 SetFont() 326 调用中提供的特定的识别符 PropertyID 以及给定的字体值。示例的 SetFont() 功能可以表现如下 :

[0030] SetFont(PropertyID, white(白色))

[0031] {PropertyID_ > SetValue(PropertyID, white) ;

[0032] }.

[0033] 熟悉本技术领域的人员会理解, 附加类因此而对于任何附加的特性提供强的类型性。当本发明支持直接调用 SetValue() 307 的应用程序时, 这超过了在编译期间提供强类型性的优点。如前所述, 强类型性允许在运行时间操作之前检测差错。在另一个实施例中, 对于附加特性 324 中的每一个, 附加类 322 提供另外一种静态方法 (例如, GetFontID() 328 (得到字体识别符))。可操作这种另外的静态方法来测试所请求的附加特性是否已经注册。因此, 可以使用这个附加静态方法 328 来保证在试图执行集和得到附加特性上的静态方法之前应用程序已经正确地注册附加特性。例如, 如果 GetFontID() 328 检测到颜色附加特性尚未注册, 则 GetFontID() 可能给出一个差错, 可能自动地注册请求对象的颜色附加特性, 等等。

[0034] 下面是提供延迟注册的一个示例实施例 :

[0035] class FontProvider (类别字体提供器)

```
[0036] {public static Font GetFont(Node n) (公共静态字体得到字体 (节
[0037] 点 n))
[0038]     {return n.GetValue(GetFontID());} (返回 n。得到值 (得到
[0039] 字体识别符))
[0040]     public static void SetFont(Node n, Font newfont) (公共静态 void
[0041] 设置字体 (节点 n,字体 新字体))
[0042]         {n.SetValue(GetFontID(), newFont);} (n。设置值 (得到字体识别符
[0043] (),新字体))
[0044]     public static DynamicProperty GetFontID() (公共静态 动态特性 得
[0045] 到字体识别符 ())
[0046]         {//Delay register Fontproperty if necessary(如果需要延迟注册
[0047] 字体特性)
[0048]             if(FontID == null) (如果 (字体识别符==零))
[0049]                 {FontID = RegisterProperty("Font", typeof(Font), "Arial");}
[0050]             (字体识别符=注册特性 ("字体", (字体) 的类型,"Arial",...))
[0051]         }
[0052]     }
[0053]     private static DynamicProperty FontID=null; (专用静态 动态特性 字
[0054] 体识别符=零)
[0055] }
```

[0056] 与其它编程模型相似,本发明支持基本类 302 的子类 (未示出)。为了简单起见,图 3 没有用图形示出任何基本类 302 的子类或任何其它附加类,因此,没有示出与另外附加类相关联的子类。然而,熟悉本技术领域的人员在阅读下面的说明书之后将能够参考其它附加类容易地设计具有子类的应用程序。一般,任何类别可以是附加特性的“提供者”,只要类别提供附加特性必需的静态方法。

[0057] 附加类 322 允许第三方开发者有能力把特性和功能动态地添加到他们的应用程序中而无需修改基本类 302。与在基本类 302 中修改特性相似,所添加的特性 (例如,第二特性集 324) 影响整个分层对象树 390 中的对象。在应用程序的执行期间,不对附加类 322 创建实例。因此,对于从基本类 302 创建实例的每个对象,相同的附加类 (附加类 322) 可以把第二特性集 324 提供给这些对象中的每一个。

[0058] 在应用程序的运行时间期间,基本对象 312 变成所创建的实例。此外,一个或多个子对象 (例如,子对象 340-350) 变成所创建的实例。在对象树 390 中的对象的创建实例可以通过编程控制 (例如, C#),通过标记语言 (例如, XML) 或通过其它手段。当通过编程控制创建对象树 390 时,编译器一般识别执行附加类 322 的指令是否有任何差错,诸如不正规的名称或数据类型。如上所述,附加类 322 提供强的类型性。因此可以在运行时间之前捕捉到差错。此外,在应用程序的开发期间,软件开发工具可以检测差错。当通过标记语言创建对象树 390 时,语法分析程序中断标记语句。一般,在标记语言中的标志的名称是相应类别的名称。

[0059] 这些子对象 340-350 中的每一个可以包括子对象 (例如,子对象 340) 中的一个或

多个子特性（例如，子特性 341）。可以把子特性的值存储在相关联的子对象中。然而，根据本发明，子对象不需要有存储值的局部存储器。此外，这些子对象 340-350 中的每一个可以包括与子对象相关联的一个或多个附加特性（例如，在虚线框中表示的附加特性 352）。如下面连同图 6 所详细描述，附加特性 352 可以从与子对象 350 相关联的局部存储器得到它的值，可以通过继承（例如，基本对象 312），通过特性页（未示出），通过程序设计方法（例如，静态方法 330 连同方法 309），以及通过根据本发明的其它手段得到它的值。在这些附加特性的值与子对象相关联的同时，这些值的存储一般不是子对象的一部分。

[0060] 再有，图 3 示出用户 - 界面环境中的编程模型。因此，人们会注意到在基本对象 312 和子对象 340-350 中的特性与对象的性能有关。基本对象 312 表示对话框，子对象 340 表示按钮对象，子对象 350 表示编辑框，而子对象 342 表示具有分别表示列表框和树的两个子对象 346 和 348 的选择器对象。这些子对象 340-350 中的每一个包括一个或多个子特性（例如，子特性 341），诸如按钮对象的按下特性以及树对象的扩展特性。

[0061] 对比之下，在附加类 322 中的特性与对象的外观有关。例如，第二特性集可以包括字体类型、颜色等的特性以及缺省值。配置这个编程模型，使之在运行时间处在子对象上设置一个特性。结果，配置每个子对象中的附加特性（例如，附加特性 352），使之接收来自附加类别的值。

[0062] 图 4 示出用于可以执行图 3 示出的编程模型的各种操作环境的几个示例语法。在每个示例语法中使用的术语“提供器”表明在语法中描述的静态方法与具有名称为“提供器”的附加类别相关联。因此，因为这些语法的每一个允许对附加类别定名，所以在两个不同的附加类别中具有相同名称的特性相互之间不会冲突。这允许开发者开发附加特性类别而无需周密考虑潜在的名称冲突。

[0063] 现在转到示例语法，如果编程环境是程序设计方法语言，诸如 Visual Basic 或 C#，则可能出现伪代码，如程序设计语法 401 中所示。程序设计语法 401 进一步包括一种方法识别符 404；参数列表，所述参数列表具有用于指定特性的名称的第一识别符 406，以及用于指定与第一识别符 406 指定的特性相关联的一个值的第二识别符 408。把参数列表封闭在括号内，并且一个周期隔开类别识别符 402 和方法识别符 404。参考图 3 示出的附加类别 322，对于附加特性“FONT”，程序设计语法 401 如下：

[0064] Attached Class. SetFont(PropertyID, value). (附加类别 设置字体

[0065] 特性识别符，值)

[0066] 感兴趣的是，这个语法显现出与以前编程模型中的语法的相似性。然而，语法不是等同的，而且程序设计语法 401 与以前的语法不同。例如，在程序设计语法 401 中，方法识别符 404 识别提供器类别（即，图 3 中的附加类 322）上的静态方法。在以前的编程模型中，方法识别符具有在提供器实例上的经识别的实例方法。因为本发明利用提供器类别（即，图 3 中的附加类 322），所以由于不需要跟踪提供器实例的寿命而编程模型可以支持在标记和特性页中的扩展器特性。程序设计语法 401 的又一个特征是语法具有较强的类型性。如上所述，由于诸如名称的拼法错误、不正确的数据类型等潜在的差错，较强类型的语法能在编译时间而不是运行时间中检测到较佳的工具支持。

[0067] 此外，对于诸如 Visual Basic、C# 等的编程环境，编程模型还允许类型性较不严格的语法（即，编程语法 411）。编程语法 411 包括识别目标对象的对象识别符 412、识别在附

加特性类别中的静态方法的方法识别符 414、包括附加特性识别符和值 419 的参数列表。附加特性识别符包括类别识别符 416 以及识别在相关联的附加类别中的附加特性的特性识别符 418。

[0068] 在另一个实施例中,可以使用诸如标记语法 421 之类的标记语言语法来实施编程模型。标记语法 421 包括标志 422、具有提供器标志符 424 以及特性识别符 426 的附加特性和一个值 428。在一个实施例中,标志 422 是目标对象(即,图 3 中的按钮对象)的名称。一般,分别用打开和关闭符号,“<”和“>”,来开始和结束标记语法 421。再有,因为附加特性不是直接在类别上,所以分析程序执行处理以查找附加特性存在于何处。

[0069] 在再另一个实施例中,可以使用具有诸如图 4 中示出的格式页语法 431 之类的格式页的可扩展的标记语言(XML)来实施编程模型。对于这个实施例,格式页语法 431 包括“标志”属性 432。标志属性 432 相应于请求处理的类别。在 XML 中的附加特性的一个示例例子如下所示:

[0070] <Button xmlns :ap =“expandoNameSpace”Pressed =“false”

[0071] ap :Expandos, String =“string”/>

[0072] 当执行根据本发明的附加特性实施的应用程序时,应用程序用具体例子说明在对象树中的对象。然后,应用程序连同特性引擎一起处理运行时间操作,所述操作包括检索和设置对象的值。图 5 和 6 示出应用程序连同特性引擎一起执行的处理。

[0073] 然而,在可以执行图 5 和 6 示出的 SetValue() 处理或 GetValue() 处理之前,必须对在这两种处理的每一种中指定的附加特性进行注册。可以在相应于附加特性(即,SetFont() 326 和 GetFont() 330)的任何静态方法中执行这个检查(例如,GetFontID() 328)。因为在每次调用任何静态方法时都必须执行这个检查,所以这个实施例招致性能损失。熟悉本技术领域的人员会理解,可以实施使特性注册最优化的其它实施例而并不偏离本发明,诸如要求应用程序来执行注册。

[0074] 附加特性的注册返回附加特性的一个唯一的 PropertyID。调用注册特性的一个例子可以按如下的形式:

[0075] BarDP = RegisterProperty(“Bar”, 0, typeof(string), typeof(Button),
0x3

[0076]);

[0077] 在这个实施例中,BarDP 变量将包附加特性的唯一的识别符。然后,在调用 SetValue() 功能和 GetValue() 功能中使用这个唯一的识别符。在一个实施例中,从物主的静态构造器直接或间接地调用 RegisterProperty。物主是指附加附加特性的类别。一旦注册了附加特性,通过调用与附加特性相关联的 Get(得到) 和 Set(设置) 静态方法,与附加特性的目标类型相匹配的任何实例就都可以“隐含地”附加特性。

[0078] 如在上述注册特性的示例调用中所示,“Bar”指定附加特性的名称。特性引擎一般不使用这个名称。然而,分析程序使用这个名称来确定是否已经对物主注册了具有相同名称的特性。目标类型是 Button(按钮)。对于附加特性,物主和目标不是相同的。缺省值是“0”。如连同图 6 所进行的详细描述,特性引擎确定在何处检索附加特性的值。因此,registerProperty 调用包括搜索特性的一个值(诸如继承、特性页)的识别阶段的性能位。在上述示例调用中,性能位“0x3”表示对局部值、特性页以及继承进行搜索以便确定附加特

性的值。因此,在一个实施例中,可以把这些性能位的解译编码到 GetValue() 静态方法中。另一方面,通过以正规方式表达特性之间的模型关系而可以提供这些阶段的确定。

[0079] 在一个实施例中,可以从所示的表 1 中检索附加类别的代码。

[0080]

```
class BarProvider:Object {  
    public static DynamicProperty BarDP=RegisterProperty(  
        "Bar", 0, typeof(string), typeof(Button));  
  
    public static string GetBar(Button button) {  
        return (string)button.GetValue(BarDP);}  
    public static void SetBar(button button, string value) {  
        button.SetValue(BarDP, value);}  
}.
```

[0081]

[0082] 表 1。

[0083] 在上述代码中,在附加类别“BarProvider”中定义名称为“Bar”的附加特性。用作为字符串的数据类型定义名称为“Bar”的附加特性,并且只可以附加到 Button 对象。静态方法是强类型的,并允许访问特性和设置特性。

[0084] 一旦已经注册了附加特性,就可以执行 SetValue() 和 GetValue() 功能。图 5 是逻辑流程图,示出根据本发明设置值的过程。在方框 501 处开始处理,其中,已经启动设置功能,使与某个特性相关联的值改变。一般,SetValue(设置值) 过程 500 将局部地存储感兴趣(此后称之为感兴趣特性) 的特性的值。在判定方框 502 处继续处理。

[0085] 在判定方框 502 处,作出是否存在感兴趣特性的局部存储器的判定。因为本发明不具备对于对象的每个实例的每个特性的存储器,所以如果以前没有存储过感兴趣特性的值,则要分配存储器(方框 504)。一旦已经分配了存储器,处理就继续进行到方框 506,如果在判定方框 502 中已经检测了局部存储器。

[0086] 在方框 506 中,把所述值复制到与感兴趣特性相关联的存储器中。一旦感兴趣特性已经改变状态,就发出状态改变的通知(方框 508)。在一个实施例中,这个通知可以包括设置一个页面重写标志位,向有关的特性指示出它们的状态不再有效。另一方面,通知可以包括向每个有关者报告源已经变化,如上述专利申请中所述。在判定方框 510 中继续进行处理。

[0087] 在判定方框 510 中,作出所述值是否存储在感兴趣特性的高速缓冲存储器中的判定。如果在感兴趣特性的高速缓冲存储器中没有值,则过程结束。然而,如果在高速缓冲存储器中有值,则使高速缓冲存储器清零(方框 512),因为这个值不再有效。过程结束。

[0088] 图 6 是逻辑流程图,示出根据本发明检索值的过程。在方框 601 处开始处理,其中,已经启动查询。在方框 602 处继续进行处理。

[0089] 在方框 602 处,应用程序指示需要更新感兴趣的特性(此后称之为感兴趣特性)

中之一。一般,这发生于当对象正在查询它们当前状态的特性时。过程 600 通过检查各个阶段而判定特性的一个值。这些阶段是在注册特性时定义的。判定方框 606-610 表示示例的阶段,但是可以添加其它阶段而不偏离本发明。在判定方框 604 中继续进行处理。

[0090] 在判定方框 604 处,过程检查高速缓冲存储器,以确定以前是否高速缓冲存储过感兴趣特性。一般,为了使检索最优化而高速缓冲存储感兴趣特性。如果以前已经高速缓冲存储过特性,则在方框 612 处继续进行过程,从高速缓冲存储器检索值。另一方面,在判定方框 606 中继续进行过程。在判定方框 606 中,作出感兴趣特性的值是否是局部的判定。如果已经执行 SetValue() 而局部地设置特性的值,则所述值是局部的,如在图 5 中所示。如果值是局部的,则在方框 612 处继续进行过程,从局部存储器中检索所述值。如果值不是局部的,则在判定方框 608 中继续进行过程。

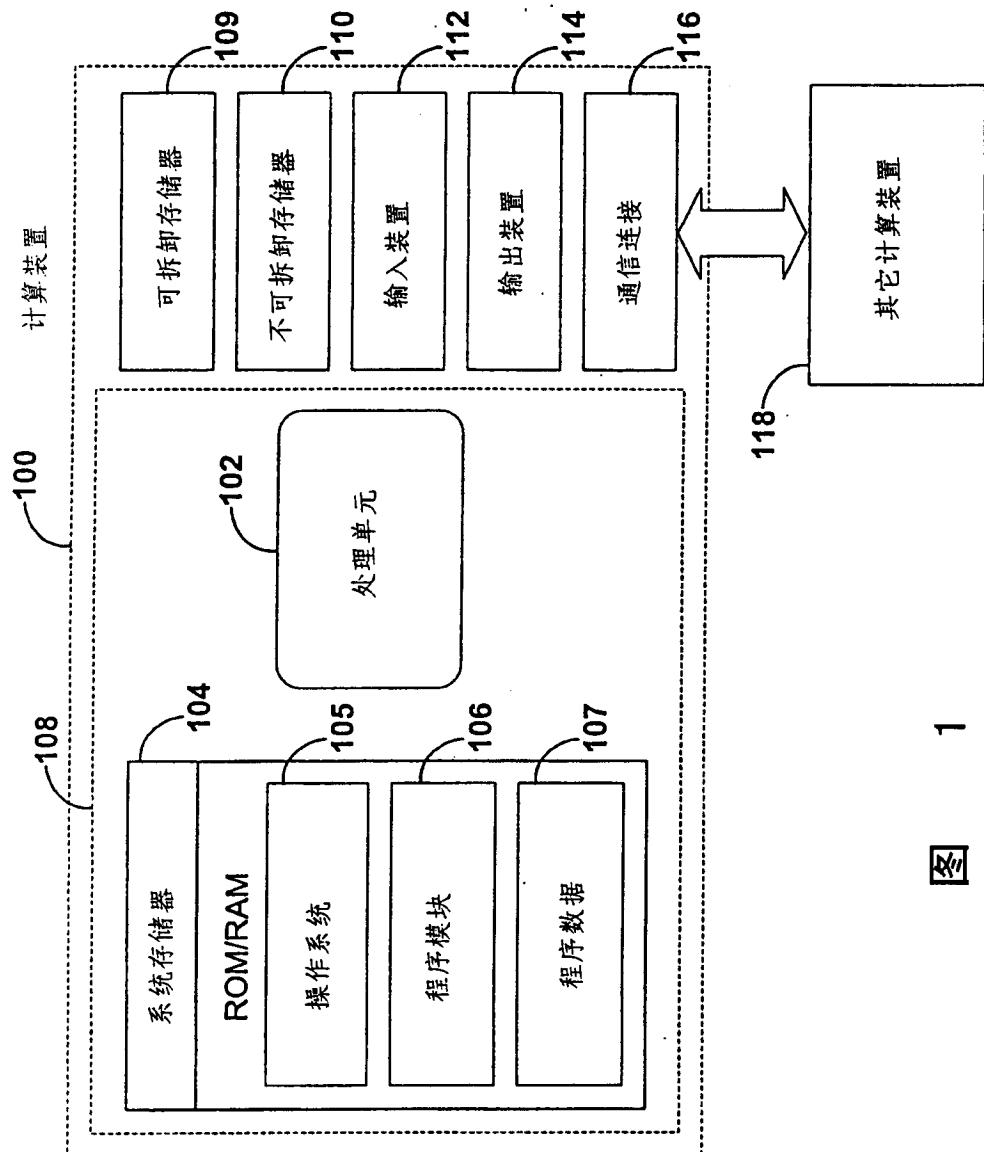
[0091] 在判定方框 608 中,作出是否可得到特性页中的感兴趣特性的值的判定。当基本类具有相关联的特性页时,特性页规定每当用具体例子说明时将如何执行基本类的每一个子类。如果可以在特性页中得到感兴趣特性,则在方框 612 中继续进行处理,从特性页检索值。然而,如果值不在特性页中,则在判定方框 610 中继续进行处理。

[0092] 在判定方框 610 中,作出是否可以通过继承(即,继承值)得到感兴趣特性的值的判定。从子对象的父对象派生继承值。如果父对象中之一具有感兴趣特性,则处理继续进行到方框 612,从父对象检索值。然而,如果没有继承值,则从与附加特性相关联的附加类别检索缺省值(方框 614)。在方框 616 处继续进行处理。

[0093] 在方框 616 处,应用程序根据接收到值的阶段计算加权量度。存储所述加权量度,并用于作出经过训练的判定,为了使将来的检索最优化,判定要高速缓冲存储哪个感兴趣特性。然后结束处理。

[0094] 此外,对于上述 SetValue 和 GetValue 过程,本发明的编程模型提供增强的功能,诸如成组查询和成组通知。因此,如所述,本发明提供一种编程模型,所述编程模型提供基于相关性的通信系统,所述系统通过使用附加类别而支持特性之间的复杂关系。使用这个编程模型构造的应用程序可以很好地定标,因为特性管理技术不需要过多的局部的、基于每个 - 实例的存储器。而是,编程模型促进特性值的再使用,诸如通过在基本类的等级使用特性页和继承。如上所述,本发明提供特性管理机制,从而在对象层次中的对象通过附加而存储特性值。

[0095] 上述说明、例子和数据提供制造和使用本发明的组成的完整的说明。由于可以制造许多本发明的实施例而不偏离本发明的精神和范围,所以本发明归属在后面所附的权利要求书中。

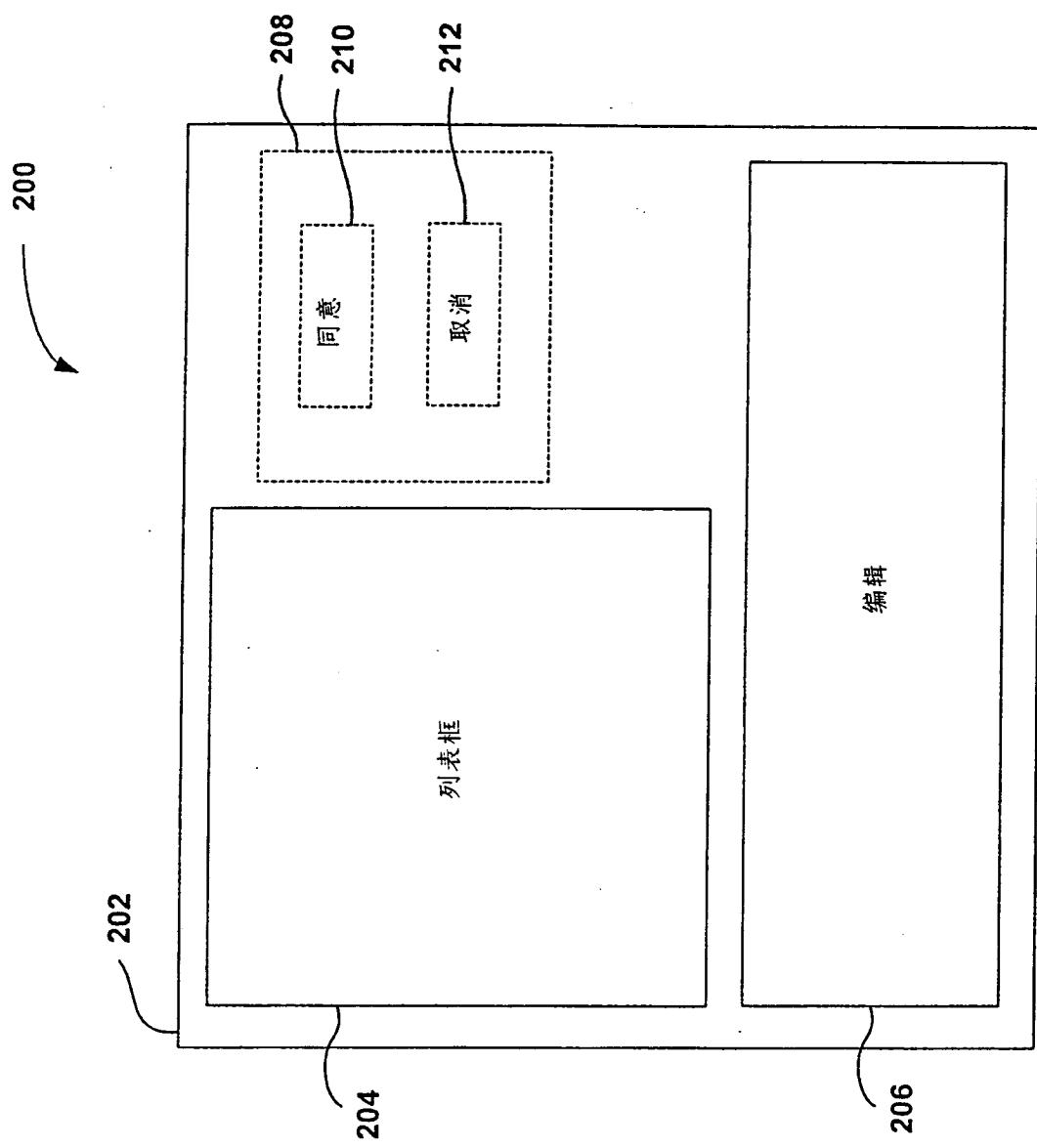


1

图

2

图



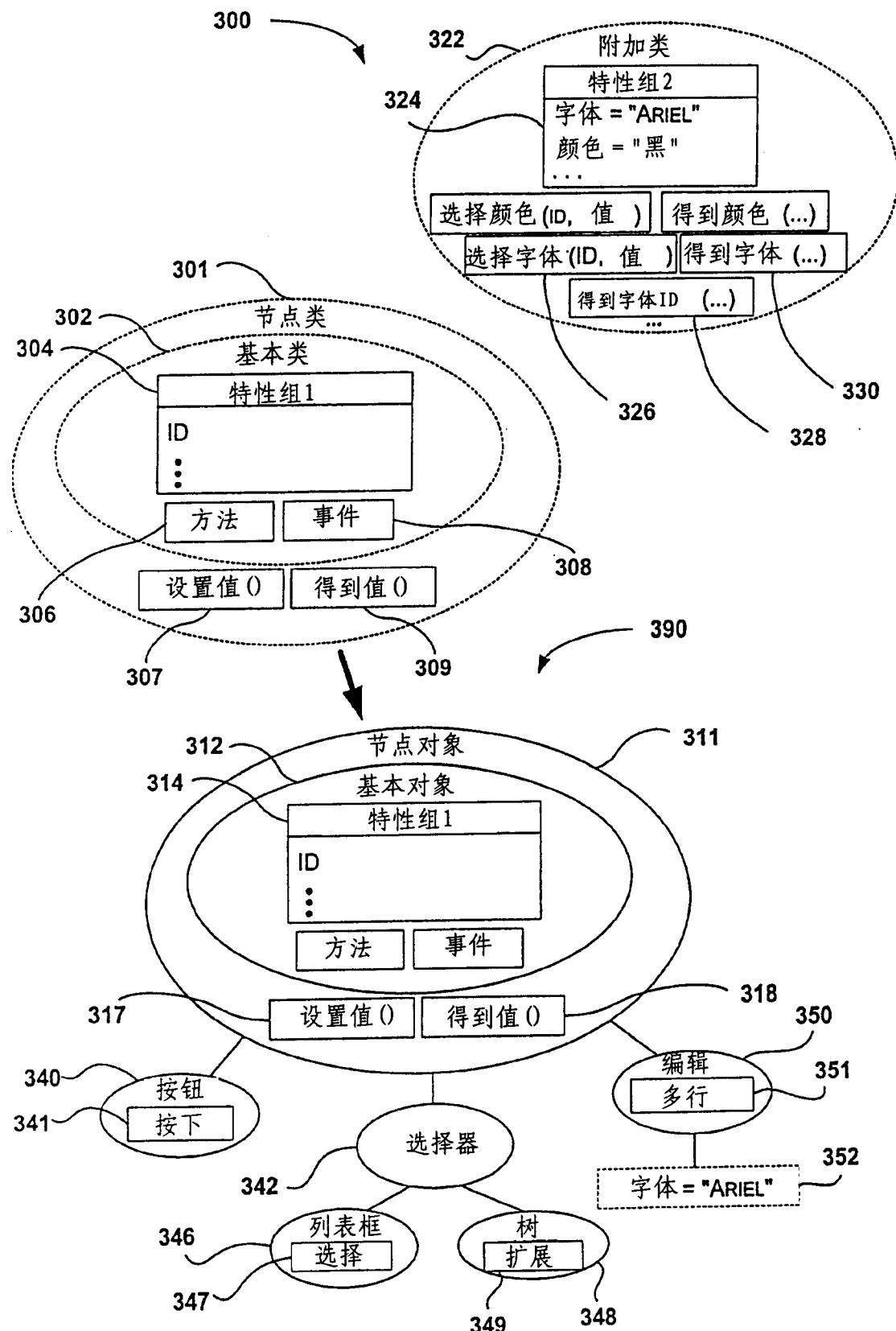


图 3

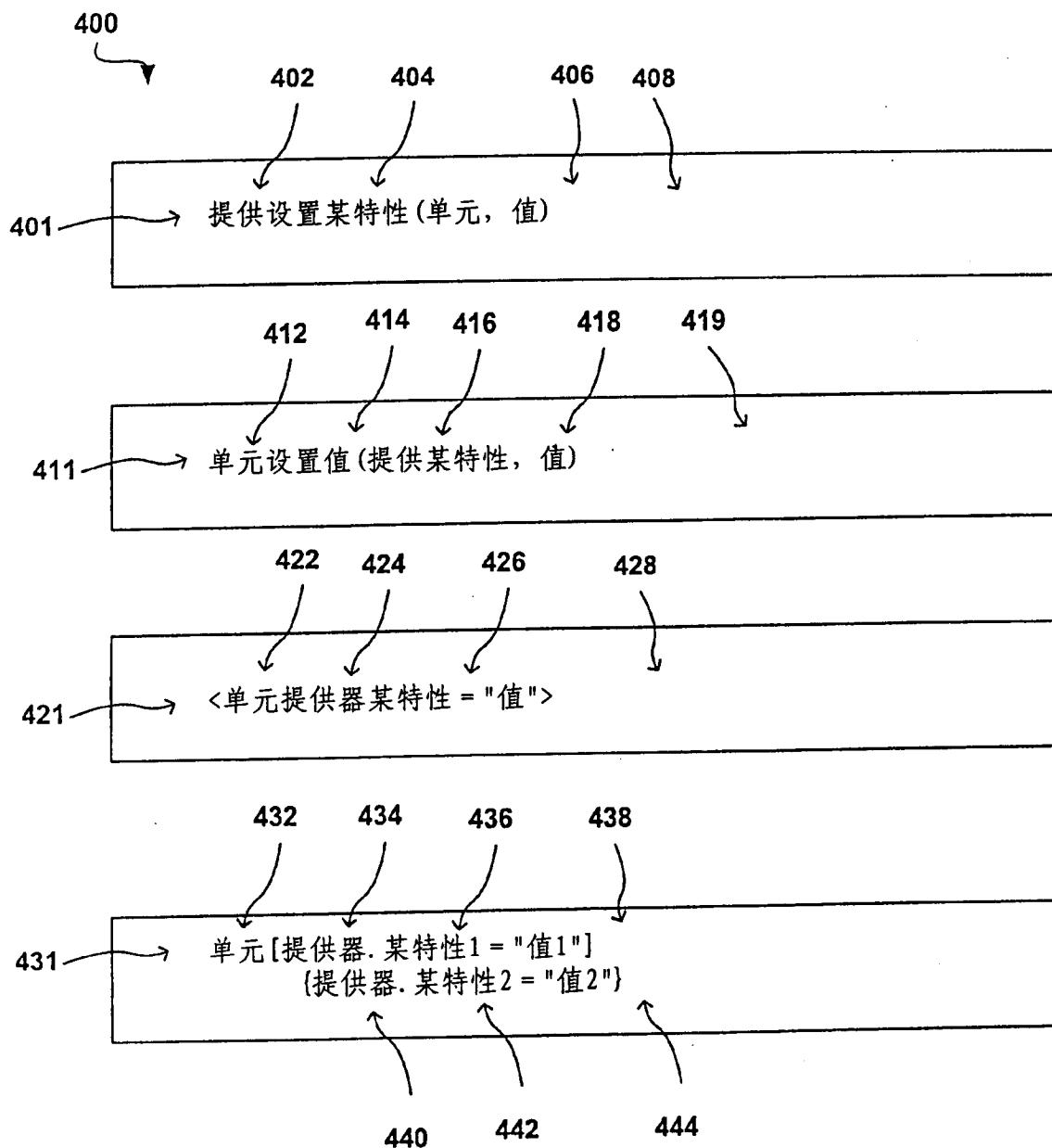


图 4

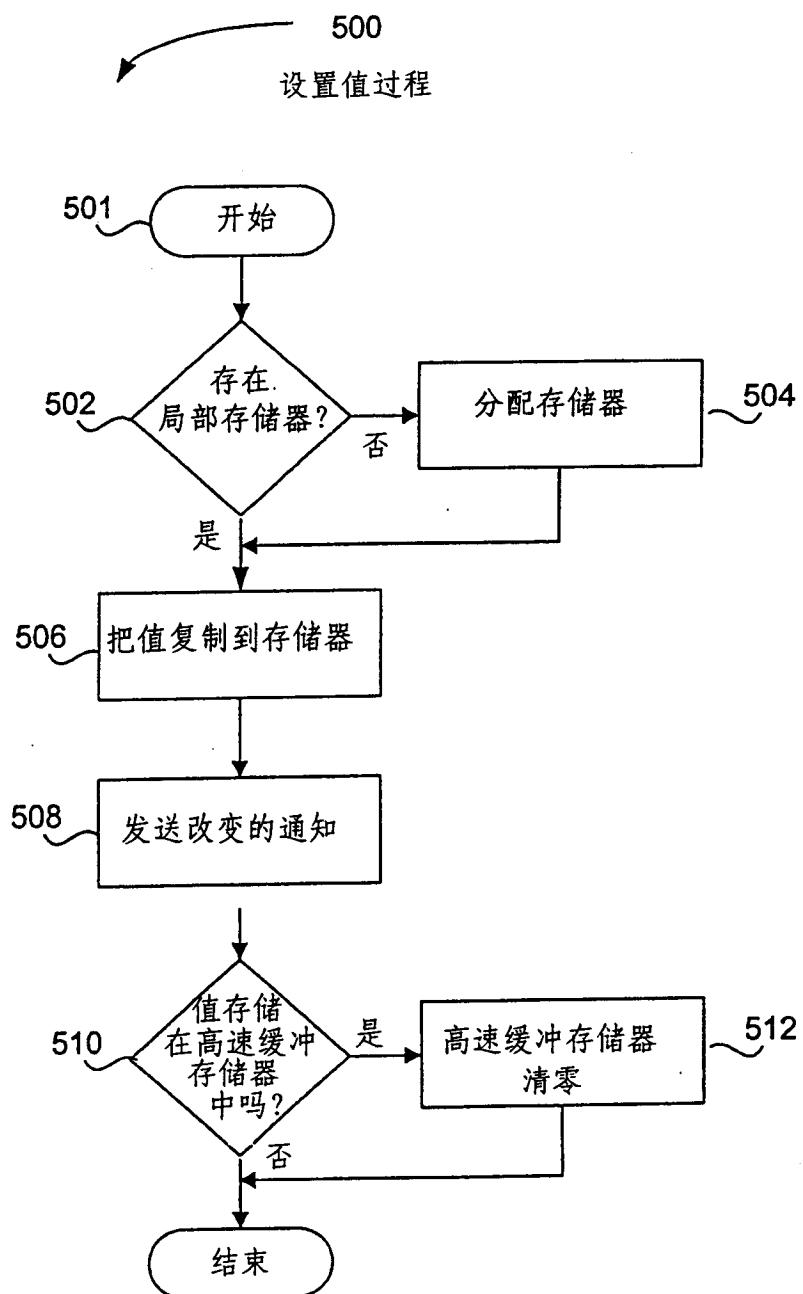


图 5

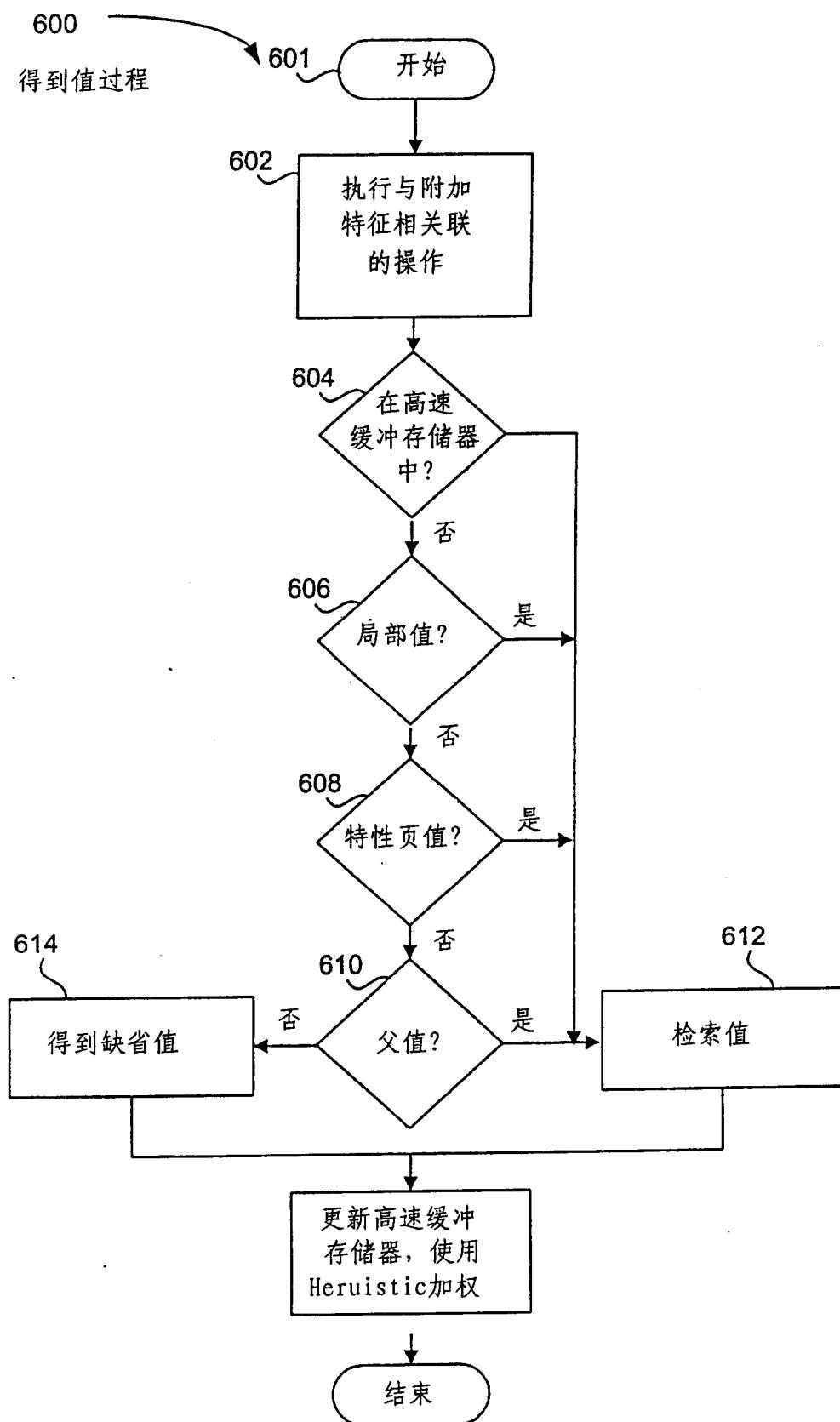


图 6