



[12] 发明专利说明书

专利号 ZL 200510128739.1

[45] 授权公告日 2009 年 11 月 18 日

[11] 授权公告号 CN 100561472C

[22] 申请日 2005.11.30

[21] 申请号 200510128739.1

[30] 优先权

[32] 2004.12.30 [33] US [31] 11/026,341

[73] 专利权人 微软公司

地址 美国华盛顿州

[72] 发明人 C·P·雅兹德泽弗司基

J·杜涅茨 O·H·弗尔

R·A·瑞尔耶

[56] 参考文献

EP 1376389 A2 2004.1.2

US 6507856 B1 2003.1.14

CN 1469247 A 2004.1.21

审查员 程小梅

[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

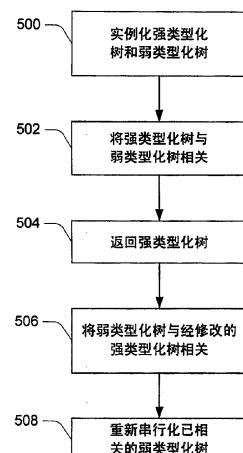
权利要求书 3 页 说明书 11 页 附图 6 页

[54] 发明名称

用于在强类型环境中保存未知标记的方法和系统

[57] 摘要

描述了用于在强类型化环境中保存未知标记的方法和系统。在一个实施例中，接收可能同时包含已知和未知的基于 XML 的元素的基于 XML 的标记。实例化与已知的基于 XML 的元素相关联的强类型化树，并实例化与已知和未知的基于 XML 的元素都相关联的弱类型化树。然后以保存未知的基于 XML 的元素的方式，将强类型化树和弱类型化树相关联。



1. 一种用于在强类型环境中保存未知的基于 XML 的元素的方法，包括：

在强类型环境中：

接收可能包含已知和未知的基于 XML 的元素的基于 XML 的标记，其中未知的基于 XML 的元素与未知的名字空间关联并且不被指定确定的类型；

实例化与所述已知的基于 XML 的元素相关联的强类型树；

实例化与所述已知和未知的基于 XML 的元素都相关联的弱类型树，包括：

对于所述基于 XML 的标记中的每个已知元素，在所述弱类型树中创建一个占位符节点；以及

对于所述基于 XML 的标记中的每个未知元素，创建保存对应于所述未知元素的未知标记的节点；以及

以保存所述未知的基于 XML 的元素的方式通过用来自所述强类型树的对象或节点来填充所述弱类型树将所述强类型树和所述弱类型树相关；

其中在所述强类型环境中，类实例属性仅能持有从基类派生的内容。

2. 如权利要求 1 所述的方法，其特征在于，所述实例化的动作是在所述基于 XML 的标记的解除串行化期间执行的。

3. 如权利要求 1 所述的方法，其特征在于，所述相关的动作是由配置成将所述基于 XML 的标记解除串行化的解除串行化器组件执行的。

4. 如权利要求 1 所述的方法，其特征在于，还包括在接收所述基于 XML 的标记以后，将所述基于 XML 的标记转换为二进制表示。

5. 如权利要求 4 所述的方法，其特征在于，所述实例化强类型树的动作包括对所述二进制表示进行操作以实例化所述强类型树。

6. 如权利要求 1 所述的方法，其特征在于，所述创建保存对应于所述未知元素的未知标记的节点的动作包括将所述未知元素的文本串与所述节点相关联。

7. 如权利要求 1 所述的方法，其特征在于，还包括在创建所述占位符节点以后，对于每个占位符节点，将各占位符节点设为所述强类型树中的对应节点。

8. 如权利要求 1 所述的方法，其特征在于，还包括向一组件返回所述强类型树，以使所述组件能够对所述强类型树进行操作以提供经修改的强类型树。

9. 如权利要求 8 所述的方法，其特征在于，还包括将所述经修改的强类型树

串行化为基于 XML 的标记。

10. 如权利要求 9 所述的方法，其特征在于，还包括在串行化期间，将所述弱类型树与所述经修改的强类型树相关。

11. 如权利要求 9 所述的方法，其特征在于，还包括在串行化期间，允许应用定义哪些节点应或不应被串行化的一个或多个规则。

12. 一种用于在强类型环境中保存未知的基于 XML 的元素的系统，包括：

在强类型环境中：

用于接收可能包含已知和未知的基于 XML 的元素的基于 XML 的标记的装置，其中未知的基于 XML 的元素与未知的名字空间关联并且不被指定确定的类型；

用于实例化与所述已知的基于 XML 的元素相关联的强类型树的装置；

用于实例化与已知和未知的基于 XML 的元素都相关联的弱类型树的装置，包括：

用于对于所述基于 XML 的标记中的每个已知元素，在所述弱类型树中创建一个占位符节点的装置；以及

用于对于所述基于 XML 的标记中的每个未知元素，创建保存对应于所述未知元素的未知标记的节点的装置；以及

用于以保存所述未知的基于 XML 的元素的方式通过用来自所述强类型树的对象或节点来填充所述弱类型树将所述强类型树和所述弱类型树相关的装置；

其中在所述强类型环境中，类实例属性仅能持有从基类派生的内容。

13. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括：

用于对所述基于 XML 的标记进行语法分析的装置；以及

用于接收已由所述用于对所述基于 XML 的标记进行语法分析的装置进行语法分析的每个标记元素的通知的装置，其被配置成实例化所述弱类型树。

14. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括：

用于对所述基于 XML 的标记进行语法分析的装置；以及

用于接收被语法分析的每个标记元素的通知的装置，并且该装置用于实例化所述弱类型树，并将所述强类型树和所述弱类型树相关。

15. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括用于对所述基于 XML 的标记进行语法分析，并将所述基于 XML 的标记转换为二进制表示的装置。

16. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括用于对所述基于 XML 的标记进行语法分析，将所述基于 XML 的标记转换为二进制表示，以及使用所述二进制表示实例化所述强类型树的装置。

17. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括用于对所述基于 XML 的标记进行语法分析，并向另一个组件返回所述强类型树，以使所述的另一个组件能够对所述强类型树进行操作以返回经修改的强类型树的装置。

18. 如权利要求 12 所述的系统，其特征在于，所述系统进一步包括用于对所述基于 XML 的标记进行语法分析，并向另一个组件返回所述强类型树，以使所述的另一个组件能够对所述强类型树进行操作以返回经修改的强类型树的装置，以及

用于将所述经修改的强类型树串行化为基于 XML 的标记，并在串行化期间将所述弱类型树与所述经修改的强类型树重新相关的装置。

用于在强类型环境中保存未知标记的方法和系统

技术领域

此发明涉及处理 XML 和基于 XML 的标记语言的方法和系统。

背景技术

通常，XML 和基于 XML 的标记语言是可扩展的。在一些系统中，可扩展性机制是基于可由处理标记的任一个任意代理实现的已知名字空间和未知名字空间的概念。未知名字空间可包含特定代理可能理解也可能不理解的任意扩展。当多个代理被链接在一起以构成管道，在管道中使标记流顺序地通过每一个代理，且代理在某种程度上修改标记时，所引入的可扩展性机制产生一个称为“标记保存”的难题——即，保存任何特定代理可能不理解的标记。

在一些系统中，已知和未知的名字空间的集合在管道中的个体代理之间可能有所不同。在这样一个系统中，任何先于另一个代理而到来的代理保存来自任何未知名字空间的标记就变得很重要，因为理解该名字空间的后续代理可能想要处理来自该名字空间的内容。

针对基于 XML 的标记语言的常规处理代理常常选择实现文档对象模型 (DOM) 树。DOM 树是弱类型的结构，它对于标记中所找到的每一个元素标记包含一个节点。因为每一个节点都是弱类型的，所以代理创建持有未知名字空间的标记的非指定类型的节点，在处理期间跳过这些节点，以及接下来将它们顺序地串行化回标记以传递给下一个代理，这是一个相当简单直接的过程。

在强类型环境中，处理未知名字空间就没有那么简单直接。更具体而言，在强类型环境中，要保存与未知名字空间相关联的标记即使不是不可能，也是很困难的，因为无法向该标记指派确定的类型。例如当 XML 语法分析器通过将标记翻译成诸如二进制形式等中间的不同形式来进一步处理标记时，可能存在更复杂的情形。在此类情况下，使用标记的代理通常不能修改此翻译步骤。

由此，此发明源自于与在强类型环境和/或涉及中间翻译阶段的环境中为标记保存问题提供解决方案相关联的考虑。

发明内容

描述用于在强类型环境中保存未知标记的方法和系统。在一个实施例中，接收可能包含已知和未知的基于 XML 的元素的基于 XML 的标记。将与已知的基于 XML 的元素相关联的强类型树实例化，并将与已知和未知的基于 XML 的元素相关联的弱类型树实例化。然后以保存未知的基于 XML 的元素的方式，将强类型和弱类型树相关。

本发明还提供了一种用于在强类型环境中保存未知的基于 XML 的元素的标记的系统，包括：在强类型环境中，其中在所述强类型环境中，类实例属性仅能持有从基类派生的内容；用于接收可能包含已知和未知的基于 XML 的元素的基于 XML 的标记的装置，其中未知的基于 XML 的元素与未知的名字空间关联并且不被指定确定的类型；用于实例化与所述已知的基于 XML 的元素相关联的强类型树的装置；用于实例化与已知和未知的基于 XML 的元素都相关联的弱类型树的装置，包括：用于对于所述基于 XML 的标记中的每个已知元素，在所述弱类型树中创建一个占位符节点的装置；以及用于对于所述基于 XML 的标记中的每个未知元素，创建保存对应于所述未知元素的未知标记的节点的装置；该系统还包括用于以保存所述未知的基于 XML 的元素的方式通过用来自所述强类型树的对象或节点来填充所述弱类型树将所述强类型树和所述弱类型树相关的装置。

附图说明

图 1 示出强类型环境的各个方面。

图 2 示出可在强类型环境中操作的个体组件。

图 3 根据一个实施例示出示例性组件。

图 4 示出示例性弱类型和强类型树。

图 5 是描述根据一个实施例的一种方法的流程图。

图 6 示出一种示例性计算设备，结合该设备可实现一个或多个实施例。

具体实施方式

概述

以下所描述的方法和系统提供可在强类型环境的上下文中保存与未知名字空间元素相关联的标记的手段。至少在一些实施例中，这是通过实例化与已知的基于 XML 的元素相关联的所谓强类型树以及与已知和未知的基于 XML 的元素相关联

的弱类型树来实现的。然后使用一种相关过程，在基于 XML 的标记的解除串行化期间将强类型树和弱类型树相关。然后由一种解除关联的过程以保存任何未知名名字空间元素的方式将基于 XML 的标记解除串行化。

以下所描述的实施例是在计算环境的上下文中描述的。各个实施例可由诸如个人计算机或 PC 等计算机执行的诸如程序模块等计算机可执行指令或代码手段实现。一般而言，程序模块包括执行特定任务或实现特定抽象数据类型的例程、程序、对象、组件、数据结构等等。

各个实施例还可在除 PC 以外的计算机系统配置中实现。例如，可在手持式设备、多处理器系统、基于微处理器或可编程的消费者电子设备、网络 PC、小型计算机、大型计算机等等中实现各个实施例。替换地或者另外地，还可在分布式计算环境中实施各个实施例，其中任务是由通过通信网络连接的远程处理设备执行的。在分布式计算环境中，程序模块可位于本地和远程记忆存储设备中。

尽管各个实施例都可被结合到以上提出的许多类型的计算环境中，以下仅在常规计算设备形式的示例性通用计算设备的上下文中描述图 6 中出现的一种示例性环境，该示例性环境将在此文档结尾处的标题“示例性计算环境”下详细描述。

强类型环境的特征

作为强类型环境的一个例子，结合图 1 考虑以下内容。假定诸如可能在计算机显示器上所显示的页等一页内容 100 包括文本 102、图形 104、图像 106、一些附加文本 108 和可点击按钮 110。此内容可被表示为（在存储器中）树 112，其根为 *FixedPage*（固定页），其节点为文本、图形、图像、文本和按钮。此树的 XML 表示在 114 示出。

在树 112 中，每一个元素都由类对象的实例表示。在此例中，*FixedPage* 是类的实例，且该类被定义为具有称为 *children*（子）的属性。*children* 属性是强类型的，且它被强类型化为某个大类的集合。该树的每一个元素都是一个类名，它们是某个基类的子类，或是从其派生而来，在此例中，该基类是 *UIElement*（UI 元素）。在此强类型环境中，*FixedPage* 类实例子属性仅可持有从基类 *UIElement* 派生的内容。

扩展的标记可能被丢失的一种方式的示例

现在考虑图 2 中所示的情形，其中应用程序 200 生成要由打印系统 204 处理

的标记 202。在此例中，打印系统 204 包括过滤器或代理 206、208 和 210 的集合。这些过滤器通常加载页或文档的标记，对其进行某种程度的修改，然后提供该标记以供下一个过滤器处理。例如，过滤器可向页添加水印，按不同顺序重新排列页，缩放、旋转、调整页上的图像，等等。

打印系统的一个例子是微软的 WinFX 系统。在 WinFX 系统中，有诸如语法分析器 212 和串行化器 214 等标准软件组件处理 XML，或在 WinFX 上下文中更准确地说，处理 XAML（可扩展应用程序标记语言）。在此类及其它系统中，过滤器 206 调用语法分析器 212 以加载标记。响应于此，语法分析器 212 向过滤器返回实例化的元素树（诸如图 1 中的树 112）。过滤器随即处理该树，对其进行某种程度的修改，并向串行化器 214 提供经修改的树。串行化器 214 随即将经修改的树串行化为经修改的标记，然后该标记被以某种方式保存，如数据存储 216 所示。下一个过滤器（在此例中是过滤器 208）随即调用语法分析器 212 以加载经修改的标记，并如上述那样对相关联的树进行操作，以继续该过程。在最后一个过滤器完成了它的处理以后，经修改的树由串行化器 214 串行化处理，并可被提供给诸如打印机等使用设备。

注意，在此特定情形中，应用程序 200 已在标记 202 中插入了元素“ext Bar”。可能此元素是要被用来以某种方式控制使用设备的。例如，如果使用设备是打印机，则此元素可能与选择用于打印该文档的特定类型的墨水相关联。但是，在强类型环境中，此元素或相关联的名字空间是未知的，因此在第一个过滤器以后将被丢失。

在强类型环境中保存扩展的标记

根据即将描述的实施例，通过实例化与已知的基于 XML 的元素相关联的强类型树以及与已知和未知的基于 XML 的元素相关联的弱类型树，强类型环境中的未知元素在处理 XML（或者在下述实现示例中为 XAML）期间得以保存。这两棵树随即被彼此相关，从而弱类型树包括未知元素，并与强类型树的已知元素相关。因为这两棵树保持相关，所以弱类型树能以保存未知标记的方式被重新串行化为 XML（或 XAML）。

作为实现示例，考虑图 3，其中提供在微软的 WinFX 平台的上下文中的一种实现。应当认识到并理解，即将描述的实现并不是被用来将要求保护的发明的应用限制于特定实现或平台。相反，提供该实现示例是为了说明如何能在这一特定环境中实现本文中所描述的各个发明原理。

在此特定示例中，系统 300 是一种基于软件的系统，它包括应用程序/过滤器 302、具有二进制转换器组件和树创建器组件的语法分析器 304、串行化器 06、定制的解除串行化器 308、以及设计者帮助程序 310。

在 WinFX 平台的上下文中，语法分析器 304 是一 WinFX 对象，它结合其它未示出的组件对基于 XML 的标记进行语法分析，将已知的元素标签与各 WinFX 类相匹配，并实例化 WinFX 类对象。语法分析器 304 返回强类型 WinFX 树。串行化器 306 是一 WinFX 对象，它取强类型 WinFX 树并将其串行化为 XML 标记。

定制的解除串行化器 308（在此上下文中也考虑定制的 XAML 解除串行化器）是一 WinFX 对象，它接收已由语法分析器 304 进行语法分析的每个标记元素的通知。定制的解除串行化器可执行任意的计算。在处理标记元素以后，它返回指示语法分析器 304 是应该使用该元素还是跳过该元素的结果代码。

设计者帮助程序 310 是一 WinFX 对象，它在强类型 WinFX 对象的标记被创建以前由串行化器 306 调用。设计者帮助程序可放出附加标记。

在操作中，系统 300 用以下方式工作。假定图 2 的 XML（或 XAML）要由系统 300 处理。注意，在图 2 的例子中，未知的 XML 元素“ext Bar”已被插入到原本为已知的 XML 元素集合中。当语法分析器 304 接收到 XML 时，在此特定情形中发生两个情况。

首先，XML 由语法分析器的二进制转换器处理，并被转换为 XML 的二进制表示，在此例中该表示称为 BAML。该二进制表示随即被保存。但是，在二进制转换过程中，依靠强类型环境中所发生的处理的力量未知的所有元素被忽略。由此，在处理 XML 中，当遇到“ext Bar”元素时，它将被忽略并且实际上在二进制转换中被丢失。

第二，语法分析器的树创建器取已保存的二进制表示，并处理该二进制表示以实例化作为强类型的对象树。这在树创建器组件的右边图示出。此树随即可能被交给应用程序/过滤器 302 进行处理。

在所示和所描述的实施例中，在二进制转换过程发生的同时，二进制转换器为 XML 中遇到的每个节点或元素调用定制的解除串行化器 308。在此例中，定制的解除串行化器将可扩展性插入到该过程中。更具体而言，对于由语法分析器 304 进行语法分析的每个节点，定制的解除串行化器都被调用，它决定应当如何处理这些节点或元素。在节点是已知的情形中（即，节点与诸如文本、图形等已知元素相关联），定制的解除串行化器 308 返回语法分析器，并指示

语法分析器照常处理该节点。另一方面，如果有未知节点或元素，则定制的解除串行化起返回语法分析器并指示语法分析器跳过该节点，并不要将其转换为二进制表示。

除了指示语法分析器 304 要跳过未知节点或元素以外，定制的解除串行化器还构建弱类型 DOM 树。即，解除串行化器从可扩展性点内构建弱类型 DOM 树。此弱类型 DOM 树在定制的解除串行化器 308 的左边图示出。

在此特定例子中，对于每个已知的元素，解除串行化器都在弱类型树中创建一个占位符节点，即，空对象。在所示例子中，这些占位符被视为小的黑色实心圆。对于未知元素（诸如“ext Bar”元素等），解除串行化器创建包含原始标记（即，标记“<ext Bar/>”的文本串）的“保存”节点。这由弱类型树中的星形节点指示。

在过程中的这个点有两棵树——一棵是包含与已知元素相关联的节点的强类型树，一棵是具有包含已知元素的占位符的节点和与未知标记相关联的节点的弱类型树。

既然两棵树都已被创建，则一个节点接一个节点地走查每一个树，以将弱类型树的节点与强类型树的节点相关。如果弱类型树中的节点有强类型对象的占位符，则弱类型树中的该节点被设为强类型树中的对应对象或节点。另一方面，如果弱类型对象中的节点未被设为强类型树中的强类型对象的占位符，则该弱类型树节点被跳过，且该过程前进至弱类型树中的下一个节点。如果弱类型树中的这下一个节点具有与强类型树中的强类型对象相关联的占位符，则该弱类型树节点被设为强类型树中的对应对象或节点，且该过程继续进行，直至两棵树都已被走查完毕。由此，此过程有效地用来自强类树的对象或节点填充弱类型树。

在相关处理以后，已完全填充的弱类型树被返回。一旦弱类型树和强类型树之间的相关已被建立，该强类型树即可被返回给应用程序 392 并可由该应用程序操作。例如，应用程序可修改或删除强类型树上的节点。

一旦应用程序已经以某种方式对强类型树进行了操作，经修改的强类型树即被提供给串行化器 306 以重新串行化为 XML (XAML)。

在重新串行化时，执行相关操作的逆操作。例如，假定应用程序 302 删除了强类型树往下的第二个节点。这在图 4 中示出。此处，串行化器找出强类型树中的哪些节点对应于弱类型树中的哪些节点，然后将弱类型树与强类型树相

关。

例如，在串行化时，串行化器 306 走查强类型树和弱类型树，并修改弱类型树上的节点以反映在强类型树上所发生的处理。在此特定例子中，串行化器遇到强类型树和弱类型树中的第一节点。如果这些节点相互对应，则串行化器移动到每棵树中的下一个节点以检查对应关系。在此特定例子中，强类型树中的下一个节点已被应用程序删除。响应于此，串行化器找出弱类型树中的对应节点并将其删除。这在图中由虚线示出。

接下来，串行化器定位强类型树中的下一个节点——此处是向下（包括了已被删除的节点）第三个节点。串行化器随即发现弱类型树中的对应节点。但是，此处在弱类型树上一个被处理的对应节点和当前的对应节点之间有附加节点（即，对应于“ext Bar”）。在此情形中，如果居于中间的节点不是对应于强类型节点的节点，则将其作为原始 XML 串行化。

串行化器随即继续处理强类型树和弱类型树，从而依照经修改的强类型树将弱类型树串行化。因此，以此方式，串行化可发生，而不与强类型环境相关联的扩展的元素得以被保存。

尽管此示例描述对弱类型树的串行化处理以将其对应于经修改的强类型树，但是应当认识到，可以用能将各种规则应用于整个串行化过程来影响对应于弱类型树的最终的串行化的 XML 的方式来使用上述方法。例如，可指定定义哪些节点应或不应被串行化的某些规则。以此方式，各种规则可驱动串行化过程，以使所生成的串行化的 XML 是期望的和可预测的形式——特别是在考虑到在强类型环境中可能有对该环境而言未知的元素被插入到 XML 中这一事实的情况下。

示例性方法

图 5 是描述根据一个实施例的一种方法的步骤的流程图。该方法可在任何合适的硬件、软件、固件或其组合中实现。在一个例子中，该方法可结合诸如上述等系统来实现。但是，应当认识到并理解，其它系统也可实现此方法，而不会偏离要求保护的发明的精神和范围。

步骤 500 实例化强类型树和弱类型树。在此例中，强类型树包含在强类型环境中已知的元素或节点，而弱类型树包含与强类型元素相关联的节点，以及与已被插入到对应的 XML 中并且在该特定的强类型环境中未必已知的元素相关联的节

点。步骤 502 将强类型树和弱类型树相关，以使弱类型树具有对应于强类型树上的已知节点的节点。

树的相关能以任何适当的方式执行。在上述过程中，弱类型树包含对应于强类型元素的节点的占位符，且相关是通过并行地走查两棵树、并为对应于强类型节点的弱类型节点插入或制造条目而发生的。

一旦树已被创建并被相关，步骤 504 即将强类型树返回给应用程序、过滤器集合或某个其它组件，以使强类型树可被操作以提供经修改的强类型树。一旦强类型树已被操作，它即被返回以与弱类型树进行相关。由此，步骤 506 将弱类型树与所返回的经修改的强类型树相关，而步骤 508 以保存所插入的未知元素的方式，将已相关的弱类型树重新串行化。

伪代码示例

以下伪代码示例提供可实现上述方法的一种方式。本领域技术人员将能理解该伪代码，它构成上述方法的一种示例性实现。应当认识到并理解，可使用实现所述实施例的其它手段，而不会偏离要求保护的本发明的精神和范围。

```

class TreeNode
{
    bool isPlaceHolderForStronglyTyped;
    object stronglyTypedObject;
    string preserveMarkupElement;
    TreeNodeCollection children;
}

class PreserveMarkupLoader:Custom.Xaml.Deserializer
{
    TreeNode weaklyTypedTree;

    PreserveMarkupLoader()
    {
        weaklyTypedTree = new TreeNode();
        //创建空树用于标记保存
    }

    SerializationAction LoadNode( reader, node )
    // 为每个经语法分析的 XML 节点调用
    {
        如果节点是来自应被保存的未知名字空间：
        • 取得未知元素的标记串(通过读入器或通过收集多个节点)
        • 向weaklyTypedTree添加新的TreeNode:
            ○ 设TreeNode.isPlaceHolderForStronglyTyped为
                false
            ○ 设TreeNode.preserveMarkupElement为所保存的标记
        • 返回SerializationAction skip
    }
}

```

如果节点是来自应由强类型运行时间环境实例化的已知名字空间：

- 向 weaklyTypedTree 添加新的 TreeNode
 - 设 TreeNode.isPlaceHolderForStronglyTyped 为 true
 - 返回 SerializationAction Normal

```

}

void PopulateTreeWithReferences( Document doc )
{
    • 并行地走查 doc 的树和 weaklyTypedTree
        ◦ 如果 weaklyTypedTree 中的 TreeNode 有
          isPlaceHolderForStronglyType==true, 则设
          stronglyTypedObject 引用来自 doc 树的对象。前移两树中的当前节
          点
        ◦ 如果 weaklyTypedTree 中的 TreeNode 有
          isPlaceHolderForStronglyTyped==false, 则只前移
          weaklyTypedTree 中的当前节点
    }
}

main()
{
    // ... 语义分析器调用
    PreserveMarkupLoader preserveMarkupLoader = new PreserveMarkupLoader();

    ParserContext pc = new ParserContext();
    用 PreserveMarkupLoader 填充 pc
    Document doc = Parser.LoadXaml( stream, pc, ... ) as Document;
    PreserveMarkupLoader.PopulateTreeWidthReferences( doc );

    // 现在我们有两个情况：
    1) doc 是强类型文档树 (或 FixedPage, 或我们想要处理的任何强类型树)
    2) preserveMarkupLoader 包含 weaklyTypedTree, 其中每个节点是以下两者中的一
       个：
        a. 对强类型树中的对象的引用节点
        b. 持有所保存的标记的节点

    以对称方式处理串行化：
    • 创建 DesignerHelper 的子类 PreserveMarkupSerializer, 它从
      preserveMarkupLoader 取 weaklyTypedTree 作为输入
    • 串行化器接收 PreserveMarkupSerializer 的实例
    • PreserveMarkupSerializer 的 SerializeNode() 方法在
      weaklyTypedTree 中找到引用当前被串行化的元素的 TreeNode 的位置。
    • 如果在该元素和前一元素之间有另一个 TreeNode, 检查该间隔所覆盖的所有
      TreeNode; 如果 TreeNode.isPlaceHolderForStronglyTyped==false,
      则将 TreeNode.preserveMarkupElement 作为原始 xml 串行化。
}

```

示例性计算环境

如以上所提及，可结合诸如在图 6 中的 620 所示等计算设备实现各个实施例。计算设备 620 包括处理单元 621、系统存储器 622、以及将包括系统存储器在内的各个系统组件耦合到处理单元 621 的系统总线 623。系统总线 623 可以是若干种类型的总线结构中的任何一种，包括存储器总线或存储器控制器、外围总线、以及使

用各种总线体系结构中的任何一种的局部总线。系统存储器包括只读存储器 (ROM) 624 和随机存取存储器 (RAM) 625。包含诸如在启动期间帮助在计算设备 620 的各元件之间传递信息的基本例程的基本输入/输出系统 (BIOS) 626 存储在 ROM 624 中。计算设备 620 还包括用于读和写硬盘 660 的硬盘驱动器 627 以及用于读或写可移动磁盘 629 的磁盘驱动器 628。

硬盘驱动器 627 和磁盘驱动器 628 分别由硬盘驱动器接口 632 和磁盘驱动器接口 633 连接到系统总线 623。各驱动器及其相关联的计算机可读介质为计算设备 620 提供计算机可读指令、数据结构、程序模块和其它数据的非易失性存储。尽管本文中所描述的示例性环境使用硬盘 660 和可移动磁盘 629，本领域技术人员将可认识到，还可在示例性操作环境中使用能存储可由计算机访问的数据的其它类型的计算机可读介质，诸如光盘驱动器和光盘、磁带盒、闪存卡、数字视频盘、贝努利盒式磁带、随机存取存储器、只读存储器、等等。

若干程序模块可被存储在硬盘 660、磁盘 629、ROM 624 或 RAM 625 上，包括操作系统 635、一个或多个应用程序 636、其它程序模块 637、以及程序数据 638。用户可通过诸如键盘 640 和定位设备 642 等输入设备将命令和信息输入到计算设备 620 中。其它输入设备（未示出）可包括话筒、操纵杆、游戏垫、圆盘式卫星天线、扫描仪等等。这些及其它输入设备常常通过耦合到系统总线的串行端口接口 646 连接得到处理单元 621，但也可由诸如并行端口 634、游戏端口或通用串行总线 (USB) 等其它接口连接。监视器 647 或其它类型的显示设备也经由诸如视频适配器 648 等接口连接到系统总线 623。除了监视器以外，计算设备通常还包括诸如扬声器和打印机 630 等其它外围输出设备，它们通过并行端口接口连接到系统总线 623。

计算设备 620 可使用到诸如打印服务器 649 等一个或多个远程计算机的逻辑连接在联网环境中工作。打印服务器 649 可以是另一个计算设备、服务器、路由器、网络计算设备、对等设备或其它普通网络节点，并通常包括以上相对于计算设备 620 所描述的许多元件，尽管通常一打印服务器 649 被专用于将来自计算设备 620 的打印请求路由到所连接的打印机 650。图 1 中所示的逻辑连接包括局域网 (LAN) 651 和广域网 (WAN) 652。此类网络环境常见于办公室、企业范围的计算机网络、内联网和因特网。

当在 LAN 网络环境中使用时，计算设备 620 通过网络接口或适配器 653 连接到局域网 651。当在 WAN 网络环境中使用时，计算设备 620 通常包括调制解调器

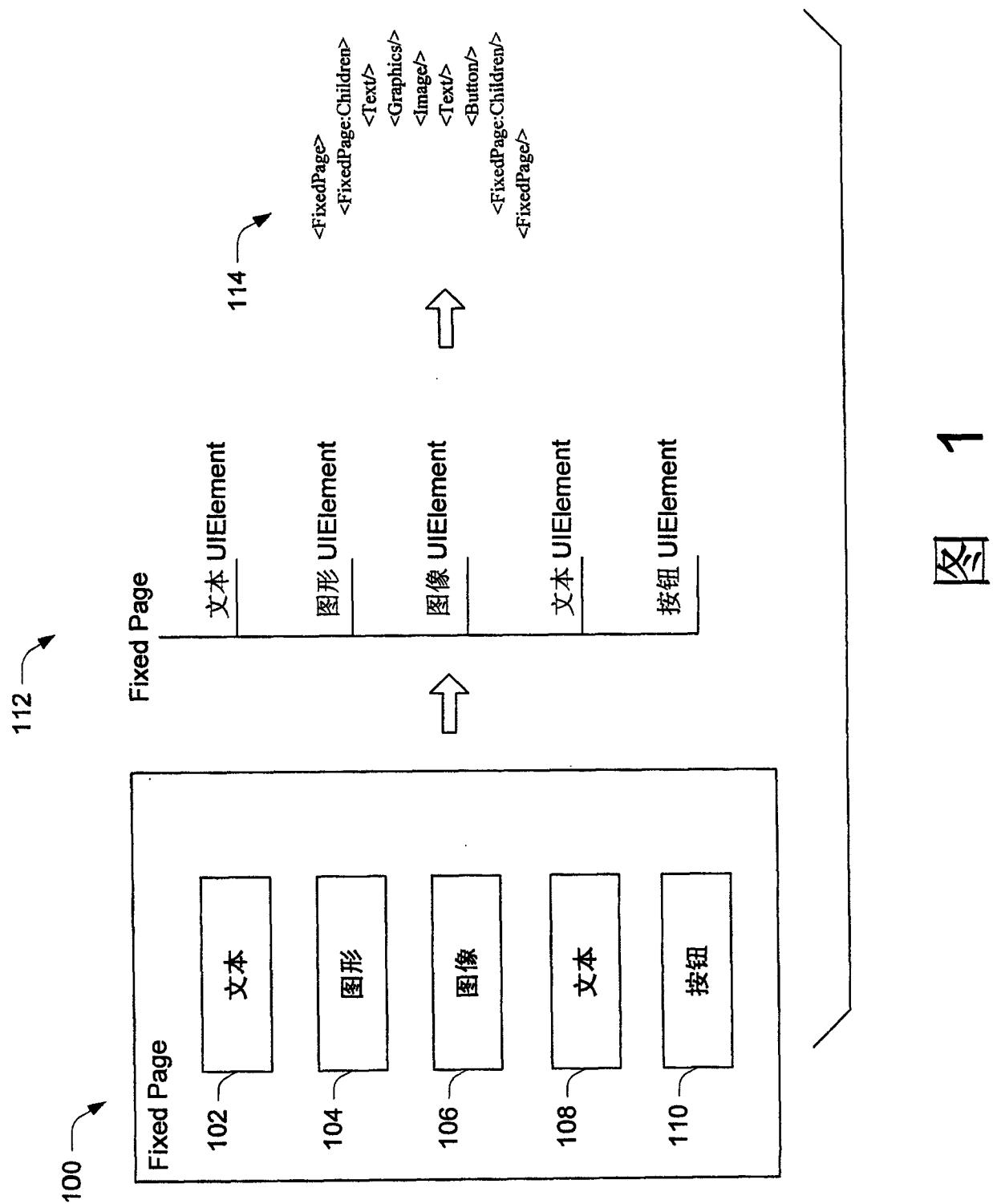
654 或用于通过 WAN 652 建立通信的其它装置。可以是内置或外置的确调制解调器 654 经由串行端口接口 646 连接到系统总线 623。在联网环境中，相对于计算设备 620 所描述的程序模块或其部分可存储在远程记忆存储设备中。可以认识到，所示网络连接是示例性的，并且可以使用建立计算机之间的通信链路的其它装置。

在以上描述中，除非另有指示，否则用对由一个或多个计算机执行动作的引用和的操作的符号表示来描述各实施例。由此，将可理解有时被称为计算机执行的这些动作和操作包括计算机的处理单元对以结构化形式表示数据的电信号的操纵。此操纵在计算机的存储器系统中的各个位置转换或维护数据，它以本领域技术人员所公知的方式重新配置或改变计算机的操作。在其中维护数据的数据结构是存储器的物理位置，它们具有由数据的格式所定义的特定属性。但是，尽管在上文中描述了各个实施例，但是并不意味着限制，如本领域技术人员将可认识到，上述的各种动作和操作还可在硬件中实现。

结论

上述的方法和系统可在强类型环境的上下文中保存与未知名字空间元素相关联的标记。至少在一些实施例中，这是通过实例化与已知的基于 XML 的元素相关联的所谓强类型树，以及与已知和未知的基于 XML 的元素相关联的弱类型树来实现的。然后使用一种相关过程，在基于 XML 的标记的解除串行化期间，将强类型树和弱类型树相关。一种解除相互关联的过程随即以保存任何未知名字空间的元素的方式将基于 XML 的标记重新串行化。

尽管是以专属于结构特征和/或方法步骤的语言描述了本发明，但是应当理解，在所附权利要求书中定义的本发明不必限于所述的具体特征或步骤。相反，揭示这些具体特征和步骤是将其作为实现要求保护的本发明的较佳形式。



202

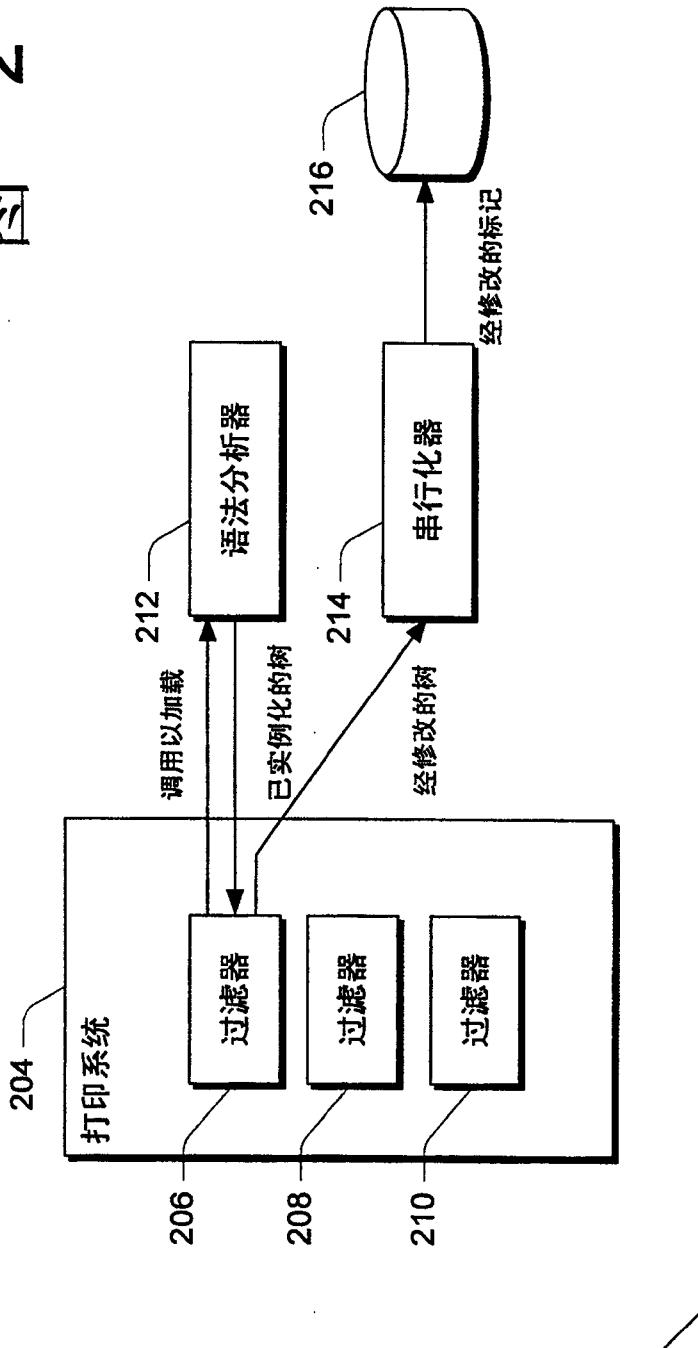
```

<FixedPage xmlns:ext = "http://foo">
  <FixedPage:Children>
    <Text/>
    <Graphics/>
    <ext:Bar/>
    <Image/>
    <Text/>
    <Button/>
  <FixedPage:Children/>
<FixedPage:>

```



图 2



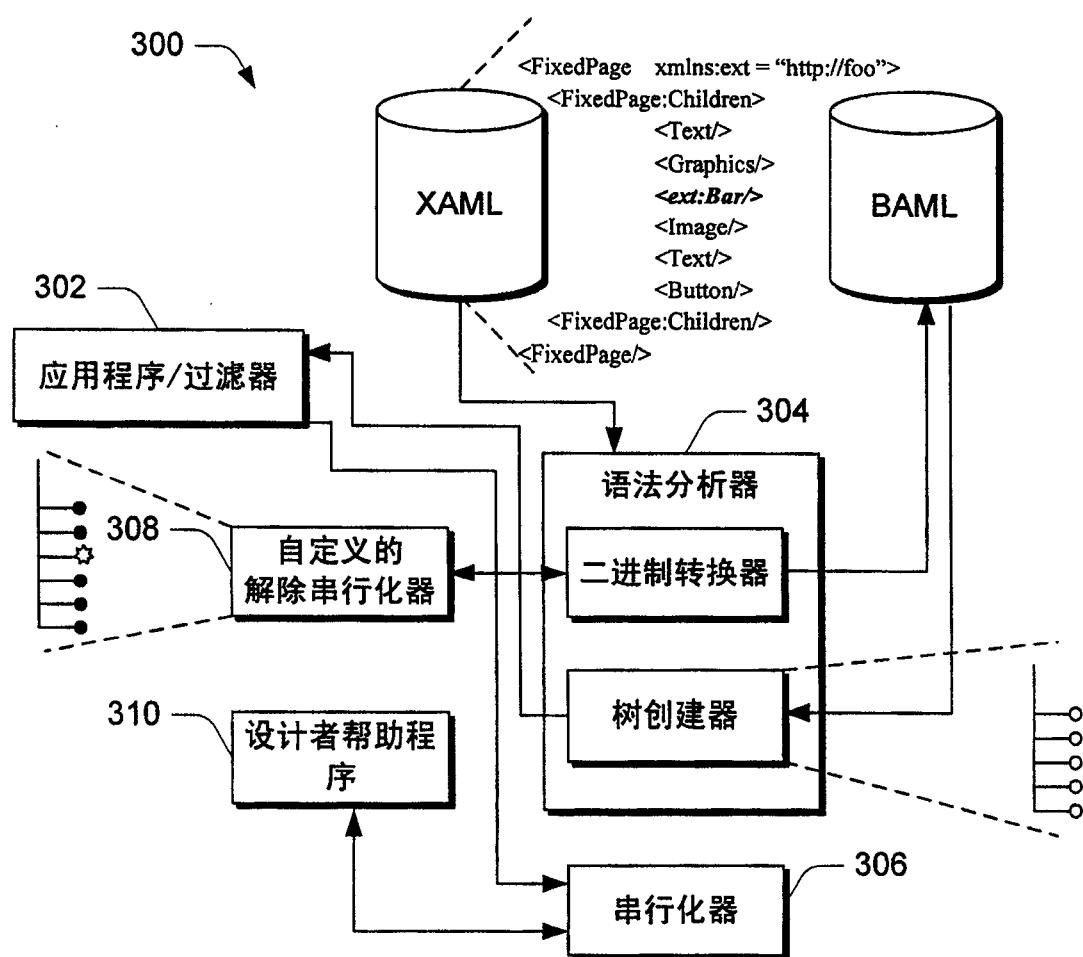


图 3

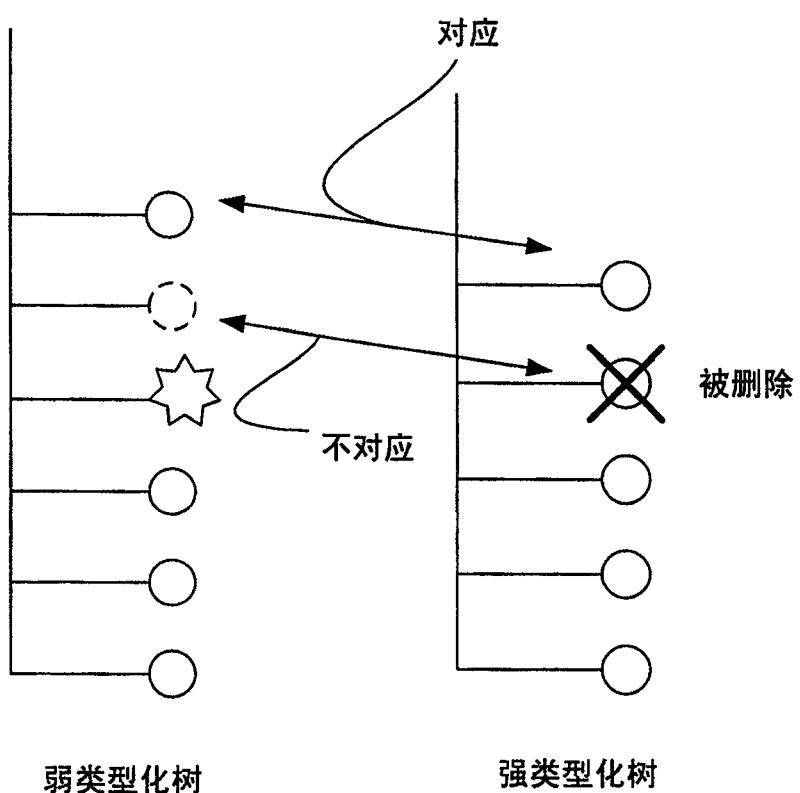


图 4

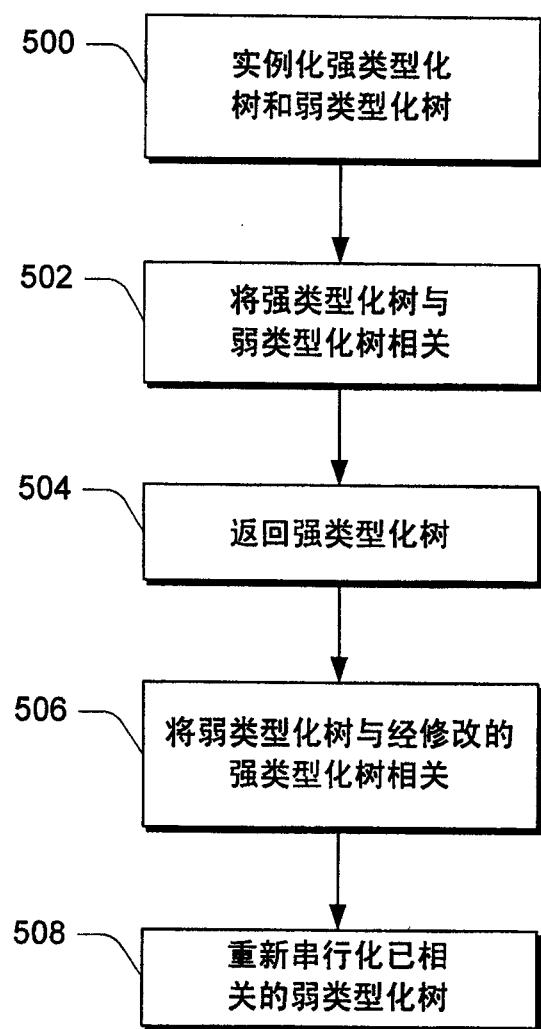


图 5

CO

