(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 101263454 B
(45)授权公告日 2011.08.17

(21)申请号 200680033200.0

(22)申请日 2006.09.12

(30)优先权数据
60/716,295 2005.09.12 US
11/335,088 2006.01.18 US

(85)PCT申请进入国家阶段日
2008.03.11

(86)PCT申请的申请数据
PCT/US2006/035691 2006.09.12

(87)PCT申请的公布数据
WO2007/033260 EN 2007.03.22

(73)专利权人 微软公司
地址 美国华盛顿州

(72)发明人 P·A·托姆普森 J·A·尼尔森
J·M·玛蒂 L·E·伯德泽斯基
D·阿斯 J·A·科帕纳
R·V·帕玛瑞斯 S·P·杰德
T·法瑞尔

(74)专利代理机构 上海专利商标事务所有限公司 31100
代理人 张政权

(51)Int.Cl.
G06F 9/44(2006.01)
G06F 17/00(2006.01)

(56)对比文件
CN 1609799 A,2005.04.27, 说明书第10页第20行至第14页第30、附图3.
US 20030084401 A1,2003.05.01, 全文.
CN 1547116 A,2004.11.17, 全文.

审查员 康健

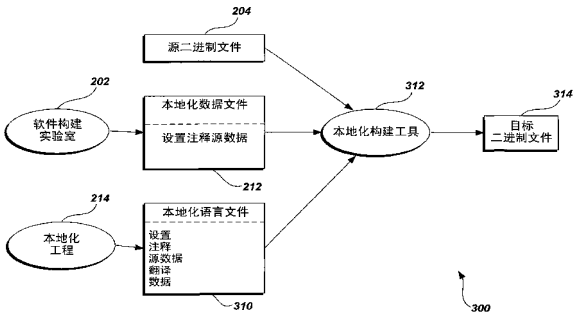
权利要求书 2 页 说明书 12 页 附图 19 页

(54)发明名称

本地化软件程序及其中包含的数据的方法

(57)摘要

描述一种方法、设备以及包括用于本地化包含在软件程序中的数据的计算机可读组件的计算机可读介质。计算机可读组件包括：由软件数据模式定义的数据元；特性库数据元，用于存储关于数据元的多个数据特性；以及自有注释数据元，包括与包含在软件程序中的数据的本地化以及具有创建、访问和操纵自有注释数据元的许可的所有者有关的信息。



CN 101263454 B

1. 一种本地化包含在软件程序中的数据的方法，包括：

用软件数据模式定义数据元；

将有关所述数据元的多个数据特性存储在特性库数据元中；以及

使用自有注释数据格式提供自有注释数据元，所述自有注释数据元包括与包含在软件程序中的所述数据的本地化以及具有创建、访问和操纵所述自有注释数据元的许可的所有者有关的信息，还包括用于通过名称指代注释的名称属性及用作指示符以指示注释是启用还是禁用的启用属性。

2. 如权利要求 1 所述的方法，其特征在于，所述软件数据模式是可扩展标记语言 XML 模式。

3. 如权利要求 1 所述的方法，其特征在于，还包括将所述用软件数据模式定义的数据元中的至少两个数据元安排成分层结构，其中所述至少两个数据元中的父数据元在所述分层结构中的位置比所述父数据元的子数据元高。

4. 如权利要求 3 所述的方法，其特征在于，所述父和子数据元的每一个包括特性库数据元。

5. 如权利要求 1 所述的方法，其特征在于，所述自有注释数据元包括针对软件本地化工具和人类操作员中的至少一个、与所述软件程序中包含的数据的本地化相关的指令。

6. 如权利要求 1 所述的方法，其特征在于，所述自有注释数据元包括对所述自有注释数据元的描述以及与所述自有注释数据元的所有权有关的信息。

7. 一种本地化包含在软件程序中的数据的方法，包括：

基于软件数据模式定义数据元，所述数据元包含在计算机可读组件集合中；

将有关所述数据元的多个数据特性存储在特性库数据元中；

提供本地化数据元的线性列表，用于将所述计算机可读组件集合分割成多个子集，分开操纵所述多个子集中的数据元，以及将所述多个子集重新合并回单个计算机可读组件集合；以及

提供自有注释数据元，包括与包含在软件程序中的所述数据的本地化以及具有创建、访问和操纵所述自有注释数据元的许可的所有者有关的信息。

8. 如权利要求 7 所述的方法，其特征在于，所述软件数据模式是可扩展标记语言 XML 模式。

9. 如权利要求 7 所述的方法，其特征在于，还包括将所述用软件数据模式定义的数据元中的至少两个数据元安排成分层结构，其中所述至少两个数据元中的父数据元在所述分层结构中的位置比所述父数据元的子数据元高。

10. 如权利要求 9 所述的方法，其特征在于，所述父和子数据元的每一个包括特性库数据元。

11. 如权利要求 7 所述的方法，其特征在于，所述自有注释数据元包括针对软件本地化工具和人类操作员中的至少一个、与所述软件程序中包含的数据的本地化相关的指令。

12. 如权利要求 7 所述的方法，其特征在于，所述自有注释数据元包括对所述自有注释数据元的描述以及与所述自有注释数据元的所有权有关的信息。

13. 如权利要求 7 所述的方法，其特征在于，所述自有注释数据元包括含有至少一个名称和相应值数据对的信息。

14. 一种本地化软件程序的方法，包括：

a) 提供包含在软件对象集合中的每一个中的数据和指令；

b) 提供本地化项目对象，包括本地化数据和以下中的至少一个：

i) 本地化项目列表对象，包含其它本地化项目对象的列表；

ii) 使用自有注释数据格式提供注释对象，所述注释对象包括有关软件程序本地化的信息，还包括用于通过名称指代注释的名称属性及用作指示符以指示注释是启用还是禁用的启用属性；

iii) 串数据对象，用于存储计算机文本信息；以及

iv) 二进制数据对象，用于存储二进制信息；

其中每个软件对象对应于由软件数据模式定义的数据结构，且每个软件对象用于访问和操纵存储在由所述软件数据模式定义的相应数据结构中的数据。

15. 如权利要求 14 所述的方法，其特征在于，所述软件对象的每一个包含采用面向对象计算机语言的软件类。

16. 如权利要求 14 所述的方法，其特征在于，所述软件数据模式是可扩展标记语言 XML 模式。

17. 如权利要求 14 所述的方法，其特征在于，所述软件对象中的至少一个读取存储在所述相应数据结构中的预选数据。

18. 如权利要求 14 所述的方法，其特征在于，所述软件对象中的至少一个将预选数据存储在所述对应数据结构中。

19. 如权利要求 14 所述的方法，其特征在于，所述软件对象中的至少两个作为父子彼此相关，其中子对象或是包含在父对象中或是从所述父对象导出。

20. 如权利要求 19 所述的方法，其特征在于，所述子对象包括指向所述父对象的返回指针。

本地化软件程序及其中包含的数据的方法

[0001] 背景技术

[0001] 近年来，软件市场日益国际化。当今，诸如文字处理、电子数据表、电子邮件等的常用软件应用程序可在不同国家使用。使软件应用程序可在不同国家使用通常使创建带有以各种本地语言（与计算机语言不同）呈现的相应用户界面以及诸如错误消息的其它人类可读文本的软件应用程序成为必要。为了增加这些软件应用程序的市场份额和市场价值，这种本地化软件应用程序的创建是必要的。包括本地语言在软件应用程序的用户界面，诸如命令界面、菜单、消息、状态信息、标签、计算结果等等中是具有主要意义的。对采用不同本地语言的软件应用程序的需要受到许多因素的驱动，包括使用不同语言且计算机日益成为日常工作和生活的一部分的国家的数量增多，使用具有需要自然语言交互而非使用会计或数学符号的技术符号交互的软件应用程序（诸如文字处理之类的办公软件应用程序）的非技术领域的数量增大，以及以自己的本地语言与软件应用程序进行交互的用户需求。本领域中用于标识以不同的本地语言创建软件应用程序的过程的通用术语是“本地化(localization)”。

[0002] 除了人类可读文本之外，可能还必需对诸如图标、色彩和形状的人类可视图形组件以及人类可听闻的声音进行本地化，以解决文化敏感性和背景。例如，在某些亚洲文化中，红色代表财富和繁荣，而在大多数西方文化中，红色代表危险或警告。所以，如果图形用户界面 (GUI) 中的对话框的符号或背景以红色显示，则这对来自不同文化的用户具有不同的混杂内涵。因此，本地化进程不止是将文本翻译成不同语言，它包括其它符号、色彩和声音等的本地化。

[0003] 对本地化软件应用程序的需求在这些软件应用程序的开发和维护过程中产生许多问题。本地化软件应用程序的开发和维护需要适当的开发工具和开发环境，用于处理和本地化各种人类可读和人类可视的软件组件。此外，软件应用程序的本地化可由多个组织完成，每个组织包括多个部门，且每个部门执行本地化进程的不同部分。当前可用的开发和组织环境的主要缺点之一是开发工具和环境所使用的数据模型的可扩展性和灵活性有限。例如，开发工具和环境所使用的某些数据是二进制格式的，这使得数据的阅读、编辑、共享和操纵十分困难。

[0004] 需要一种数据格式来提供跨不同组织和开发工具的一致性、可扩展性和灵活性。此外，需要用于访问和操纵这些数据的标准功能和数据接口方法。

[0006] 发明内容

[0005] 提供本概述是为了以简化形式引入概念精选，这些概念将在以下详细描述中得到进一步描述。本概述不旨在标识要求保护主题的关键特征，也不旨在用于帮助确定要求保护主题的范围。

[0006] 一种本地化包含在软件程序中的数据的方法，包括：用软件数据模式定义数据元；将关于数据元的多个数据特性存储在特性库数据元中；以及使用自有注释数据格式提供自有注释数据元，所述自有注释数据格式包括与包含在软件程序中的数据的本地化以及具有

创建、访问和操纵自有注释数据元的许可的所有者有关的信息，还包括用于通过名称指代注释的名称属性及用作指示符以指示注释是启用还是禁用的启用属性。

[0007] 还描述了一种本地化包含在软件程序中的数据的方法，包括：基于软件数据模式定义数据元，所述数据元包含在计算机可读组件集合中；将关于数据元的多个数据特性存储在特性库数据元中；提供本地化数据元的线性列表，可用于将计算机可读组件集合分割成多个子集，分开操纵多个子集中的数据元以及将多个子集重新合并成单个计算机可读组件集合；以及提供自有注释数据元，包括与包含在软件程序中的数据的本地化以及具有创建、访问和操纵自有注释数据元的许可的所有者有关的信息。

[0008] 进一步描述了一种用于本地化软件程序的方法，包括：提供包含在软件对象集合中的每一个中数据和指令；提供本地化项目对象，包括本地化数据和以下中的至少一个：包含其它本地化项目对象列表的本地化项目列表对象；使用自有注释数据格式提供注释对象，所述注释对象包括关于软件程序本地化的信息，包括用于通过名称指代注释的名称属性及用作指示符以指示注释是启用还是禁用的启用属性；串数据对象，用于存储计算机文本信息；以及二进制数据对象，用于存储二进制信息，其中每个软件对象对应于由软件数据模式定义的数据结构，且其中每个软件对象用于访问和操纵存储在由软件数据模式定义的相应数据结构中的数据。

[0009] 附图说明

[0010] 通过结合附图参考以下详细描述，本发明的以上诸方面以及许多附加优点将变得更容易领会和更好地理解，在附图中：

[0011] 图 1 是一示例性本地化进程的示意图；

[0012] 图 2 是示出一示例性本地化数据流的示意图；

[0013] 图 3 是示出另一示例性本地化数据流的示意图；

[0014] 图 4 是示出对 XML 数据文件进行示例性文件分割和合并操作的示意图；

[0015] 图 5A 是一示例性特性包的示意图；

[0016] 图 5B 是一示意性布尔 XML 元素的示意图；

[0017] 图 5C 是一示意性整数 XML 元素的示意图；

[0018] 图 5D 是一示意性串 XML 元素的示意图；

[0019] 图 5E 是一示例性 XML 串 XML 元素的示意图；

[0020] 图 6 是带有源属性的一示例性注释数据格式的示意图；

[0021] 图 7 是带有注释的一示例性本地化项目数据格式的框图；

[0022] 图 8 是一示例性自有注释数据格式的示意图；

[0023] 图 9A 是一示例性设置数据格式的示意图；

[0024] 图 9B 是一示例性枚举元素的示意图；

[0025] 图 9C 是一示例性列表元素的示意图；

[0026] 图 9D 是一示例性选项表元素的示意图；

[0027] 图 10 是一示例性本地化 XML 数据格式的示意图；

[0028] 图 11A 是包含在 CDATA XML 元素中的文本数据的一示例性示意图；

[0029] 图 11B 是包含在 CDATA XML 元素中包括括号字符的文本数据的一示例性示意图；

[0030] 图 12 是本地化模式与对应对象模型的一示例性关系的框图；

[0031] 图 13A 是带有返回指针的一示例性父对象和子对象的框图；

[0032] 图 13B 是带有返回指针和文件指针的一示例性父对象和子对象的框图；

[0033] 图 14 是带有外部定制文化信息的一示例性对象模型的示意图；

[0034] 图 15 是部分加载数据的一示例性方法的功能流程图；

[0035] 图 16 是用回调功能细化数据加载的一示例性方法的功能流程图；

[0036] 图 17A 是部分保存数据的一示例性方法的功能流程图；

[0037] 图 17B 是合并数据保存的一示例性方法的功能流程图；

[0038] 图 18 是细化地存储数据的一示例性方法的功能流程图；

[0039] 图 19 是一示例性本地化对象模型的示意图；

[0040] 图 20 是创建和提供本地化数据文件的一示例性方法的功能流程图；

[0041] 图 21 是向本地化项目添加注释的一示例性方法的功能流程图；

[0042] 图 22 是文件剥离的一示例性方法的功能流程图。

[0043] 详细描述

[0044] 描述了一种用于定义标准可扩展本地化数据以及用于访问和操纵数据的对象模型的系统和方法。虽然该系统和方法非常适用于本地化进程，但是该系统和方法可在其它软件环境中使用，其中涉及共享同一底层数据的多个开发工具和组织。因此，应该理解，本发明不应被解读为限制于在本文所述的示例性实施方式中的应用，并且这些示例性实施方式不应被解读为是限制性的。

[0045] 图 1 是一示例性本地化进程 100 的示意图。示例性本地化进程包括具有多个不同阶段的操作循环。这些阶段包括开发阶段 102、本地化阶段 104、翻译阶段 106 以及构建阶段 108。本领域技术人员将会意识到，操作循环阶段可包括比在本示例实施方式中描述的更少或更多的阶段。例如，可将某些阶段合并以产生更少的阶段或者进一步分解以产生更多的阶段。在开发阶段 102 中，软件应用程序（软件应用程序）的开发工程师开发软件代码和用户界面 (UI)。UI 可包括文本、可视和可听闻组件。例如，开发工程师为软件应用程序编写和编译代码。代码可采用诸如 C、C++、C# 等多种可用编程语言中的任一种来编写，并且包括源代码文件、报头文件和资源文件。资源文件通常包含可视和其它 UI 元素，诸如位图。开发工程师还可在这些文件中诸如资源文件的某些文件中包含注释。注释包括关于代码或 UI 元素、以及对在本地化软件应用程序的开发过程中使用的诸如各种软件编译器、管理工具和构建工具之类的软件工具的指令的评论。软件工程师将包括已编译代码和资源文件的文件集合传递到本地化阶段 104，在此本地化工程师继续进行本地化进程。本地化工程师向这些文件添加更多的注释，这可应用于所有目标语言和文化。然后，这些文件被传递到翻译阶段 106，在此针对每种特定语言对软件应用程序进行翻译。最终，这些文件被传递到构建阶段 108，在此构建文件以对多种语言中的每一种产生可执行软件应用程序文件。

[0046] 图 2 是示出从开发阶段 102 到本地化阶段 104 的一示例性本地化数据流 200 的示意图。本地化解析器工具 210 用于合并多个文件，包括源二进制文件 204、注释文件 206、和设置文件 208，并且产生输出本地化数据文件 212。源二进制文件 204 来自开发工程师和软件构建实验室 202。本地化文件 212 在本地化阶段 104 被 传递到本地化工程 214 的文件。在一个示例性实施方式中，本地化数据文件 212 包括来自设置文件 208 的设置、来自注释文件 206 的注释、以及来自源二进制文件 204 的源二进制数据。由软件构建实验室 202 提供

的源二进制文件 204 包括从采用诸如英语的原始语言构建软件应用程序源文件过程中得到的二进制数据。

[0047] 对嵌入到注释文件 206、源二进制文件 204 和设置文件 208 中的注释加标签以表明注释的所有者和来源。例如，可用“DEV”对来自开发工程师的注释加标签，以表明由该标签指示的注释来源。在本地化进程中使用的诸如注释提取器工具的各种软件工具也可以向软件应用程序文件添加注释。例如，注释提取器工具可用“RCCX”对由注释提取器工具拥有的注释加标签。如果在注释提取器工具所操作的输入文件中不存在注释，则注释提取器工具可以不产生输出文件。注释是大小写敏感的，其中小写和大写字母定义了不同的单词，或者注释不是大小写敏感的。注释还可被启用或禁用。例如，本地化管理工具可用“LCI”对注释加标签，并且可用于禁用其它 DEV 和 RCCX 注释。软件应用程序构建工具拥有的特定类型注释中的一种或多种，这类注释由代表构建工具进程的诸如 DEV 和 RCCX 的特定标签标识。在一个示例性实施方式中，诸如软件构建工具的工具行为受在配置文件中设定的参数的控制。

[0048] 两种不同类型的文件包含并声明同一类型注释中的一个或多个的所有权，如注释标签所定义的。多个文件对同一类型注释的所有权声明会产生冲突。这种所有权冲突可通过例如将所有权赋予最近文件来解决，或者这种冲突可基于预先赋予文件的所有权优先级来解决。因此，较高优先级的文件将对所有权发生冲突的注释类型具有更好的声明。也可使用其它类型的冲突解决方案。因此，这些示例应该被解读为是示例性而非限制性的。

[0049] 当包含不同类型注释的多个文件被合并时，如果所有权发生冲突则发出警告或错误消息。如果在注释合并操作期间遇到有意更改，则发出表明该情况的信息消息。例如，当由于自有注释的最近版本文件可用而忽略该注释时，可发出信息消息。如果注释无法被禁用，则生成警告消息。类似地，如果并非由一文件或工具拥有的注释被禁用，则发出警告消息。在一个示例性实施方式中，将注释类型的所有权从现有所有者转让给新的所有者。例如，将 DEV 类型注释的所有权从解析器工具传递到构建工具。在一个示例性实施方式中，每个所有权都具有该所有者所拥有的注释类型的所有权列表。如果将未在所有者的所有权列表上的新注释类型赋予该所有者，则该所有者保留该所有权并发出警告消息。在一个示例性实施方式中，如果两 个文件声明同一注释的所有权，则发出错误消息。这种所有权冲突可在本地化进程的诸如构建阶段 108 的后续阶段中得到解决。在一个示例性实施方式中，不包含资源文件拥有的注释类型的资源被视为具有空的启用注释，并且在注释合并操作过程中如此处理。

[0050] 如上所述，在本地化进程中，可向软件源代码和本地化文件添加注释以向本地化进程中的后续步骤提供信息和指令。注释帮助软件的本地化人员改进质量并降低本地化成本。提供注释的益处包括信息共享、帮助创建伪构建、以及检验翻译完整性。在一个示例性实施方式中，共享信息包括提供关于串资源的标准本地化指令，这减少了由这种串资源的不正确本地化所产生的错误。串资源包括向软件用户呈现的文本消息，诸如警告。伪构建是软件的临时测试构建（即软件代码的编译），由测试团队用来在产品开发周期的早期找到本地化错误，以便计划实际本地化构建的测试以及降低总体的本地化构建成本。通过使用本地化指令和注释来检验翻译完整性。通过将本地化人员提供的翻译与对本地化的限制集进行匹配来检验翻译完整性。限制集包括对软件原始语言与对软件进行本地化所针对的

目标语言之间的诸如词语和短语的信息进行匹配。

[0051] 在一个示例性实施方式中，使用注释提取器工具。如上所述，注释提取器工具是在一种本地化工具，它在包括待提取注释的文件上运行并将这些注释编写到诸如本地化数据文件的输出文件。在另一示例性实施方式中，本地化进程中所使用的每个工具都可生成注释并对这些注释加标签以将该工具标识为注释的源。工具可用指示不同源的源标签生成注释。例如，注释提取器工具可生成“DEV”注释。在这种情形中，在具有相同标签却来自不同源的两个注释之间可能发生抵触。在一个实施方式中，注释盖写模型用于禁用抵触注释。在处理期间忽略禁用注释。

[0052] 图 3 是示出从本地化阶段 104 和翻译阶段 106 到构建阶段 108 的另一示例性本地化数据流 300 的示意图。在本示例中，本地化构建器 312 对源二进制文件 204、本地化数据文件 121 和本地化语言文件 310 中包含的数据进行处理以产生输出文件的目标二进制文件 314。如上相关于图 2 所述，本地化数据文件 212 包括设置从其它文件中的数据结合的设置、注释和源数据。本地化语言文件 310 包括由本地化工程 214 添加的设置、注释、源数据和翻译数据。本地化构建工具 312 使用源二进制文件 204、本地化数据文件 212 和本地化语言文件 310 作为输入文件并产生目标语言的目标二进制文件 314。目标二进制文件 314 的创建是最终的本地化软件应用程序产品，并且是本地化进程的主要目的。

[0053] 以上描述的工具和进程依赖于一致的通用数据格式，这些工具基于该格式以标准化方式集成和处理数据。在一个示例性实施方式中，本地化可扩展标记语言 (XML) 模式用于定义一致的数据格式，以供上述各种本地化工具和相关文件使用。本地化 XML 模式提供允许不同小组和组织开发处理特定任务的软件工具的可扩展 XML 格式。本地化 XML 模式还允许开发可在多个组织之间共享的工具和数据，进而促进小组间合作。例如，本地化解析工具 210、本地化构建工具 312 和本地化管理工具可在本地化进程中对不同的文件使用和共享同一数据格式。在一个示例性实施方式中，本地化 XML 模式还可以是可扩展的。本地 XML 模式的可扩展性允许其它各方开发带有新特征的新工具而无需更改数据格式。

[0054] 图 4 是示出基于可扩展本地化 XML 模式分别对本地化 XML 数据文件 404 进行示例性文件分割和合并操作 400 和 402（与注释合并操作不同）的示意图。在本示例性实施方式中，使用文件分割操作 400 将本地化 XML 数据文件 404 分割成多个部分数据文件 406。部分数据文件 406 可由多个组织或并行处理软件工具并行使用，以便独立于其它部分数据文件 406 处理每个部分数据文件 406。例如，开发多个软件工具的数据文件的多个第三方中的每一个可分别使用与由多个第三方中的该方开发的软件工具相关的一个部分数据文件。作为另一示例，将同一软件应用程序翻译成多种语言的多个组织可使用由文件分割操作 400 创建的适当部分数据文件 406 来创建已翻译版的软件应用程序资源。当多个组织完成部分数据文件 406 的处理时，使用文件合并操作 402 将部分数据文件 406 合并成单个本地化 XML 数据文件 408。

[0055] 本地 XML 数据文件 404 包括指定本地化信息的 XML 元素。包含在本地化 XML 数据文件 404 中的这些元素之一是特性包。图 5A-5E 示出特性包和相应 XML 元素的示例性实施方式。图 5A 是一示例性特性包数据结构 500 的示意图。特性包 502 是用于存储任意数量特性的数据容器。在一个示例性实施方式中，每个复杂数据类型都与至少一个特性包相关联。复杂数据类型是包含其它数据类型的数据类型。例如，包含其它 XML 元素的 XML 元素

是复杂数据类型。在一个示例性实施方式中，使用“名称”属性定义的唯一名称和值被赋予该特性。值必须是由本地化 XML 模式支持的数据类型。在本地化 XML 模式中定义的每个复杂数据类型包括存储由本地化 XML 模式的使用者要求的任意数量数据的特性包元素。图 5A 所示的示例性特性包 502 包括布尔数据类型 504、整数数据类型 506、串数据类型 508 和 XML 数据类型 510。

[0056] 图 5B 是一示例性布尔 XML 元素 522 的示意图。布尔 XML 元素 522 包括属性列表 524。XML 中的数据类型属性用于表示关于数据类型的信息，诸如数据类型的名称和值。属性列表 524 中包含的属性之一是名称 526。在一个示例性实施方式中，名称属性 526 是 α 数字串。值 528 是布尔 XML 元素 522 的另一属性。值 528 表示布尔 XML 元素 522 的逻辑值。如本领域已知的，逻辑值包括真和假两种逻辑状态值。

[0057] 图 5C 是一示例性整数 XML 元素 542 的示意图。整数 XML 元素 542 包括属性列表 544。属性列表 544 中包含的属性之一是名称 546。在一个示例性实施方式中，名称属性 546 是 α 数字串。值 548 是整数 XML 元素 542 的另一属性。值 548 表示整数 XML 元素 542 的整数值。

[0058] 图 5D 是一示例性串 XML 元素 562 的示意图。串 XML 元素 562 包括属性列表 564。属性列表 564 中包含的属性之一是名称 566。在一个示例性实施方式中，名称属性 566 是 α 数字串。值 568 是串 XML 元素 562 的另一属性。值 568 包括含有 α 数字以及其它字符的字符串，由串 XML 元素 562 表示。

[0059] 图 5E 是一示例性 XML 串 XML 元素 582 的示意图。XML 串 XML 元素 582 表示任何有效的 XML 语句。XML 元素 582 包括属性列表 584。属性列表 584 包含名称属性 586。在一个示例性实施方式中，名称属性 586 是 α 数字串。属性列表 584 还包括任何 XML 语句属性 588，包括任何有效的 XML 语句。

[0060] 本领域技术人员将会意识到，属性包元素 502 的其它变体也是有可能的。例如，属性包元素 502 可包括一类数据元，诸如“任何 (Any)”元素（未在以上图中示出），其中任何 (Any) 元素包括名称属性、类型属性和值属性。在这种示例性实施方式中，类型属性指定如何解释值属性。例如，类型可以是“Unsigned Integer（未赋值整数）”，且值可以是“15”。

[0061] 图 6 是带有源属性的一示例性注释数据格式 600 的示意图。在一个示例性实施方式中，注释元素 603 包括为人类操作员提供有关本地化进程信息的自然语言文本以及向处理注释文件 206 和本地化 XML 数据文件 408 的人类操作员和软件工具提供作为指令的预定文本串。图 6 所示的注释元素 602 包括属性列表 604。在一个示例性实施方式中，属性列表 604 包括名称属性 606、启用属性 608、和 SRC（对于源）属性 610。名称属性 606 用于通过名称指代注释。启用属性 608 用作指示符以指示注释 602 是启用还是禁用。SRC 属性 610 指示注释的源，即 SRC 属性 610 是标识注释的所有者和源的标签。如上参照图 2 所述，不同类型的注释由不同所有 者拥有。注释所有者可包括有关与注释所有者的责任范围相关的本地化进程的指令和其它信息。例如，软件开发者可通过注释形式提供一般信息和指令。对由特定所有者作出的注释加标签，以标识该注释的所有者。在一个示例性实施方式中，由每个所有者作出的注释仅由这些注释所属的所有者来操纵。在另一示例性实施方式中，可将注释的所有权从一个所有者传递到另一个所有者。例如，可以允许标有“DEV”（即开发者，如上所述）的注释由注释提取器工具拥有，该工具通常只拥有标为“RCCX”的注释。在一示

例性实施方式中，注释名称和注释可以是大小写不敏感的。注释还可被启用和禁用。如上所述，在处理过程中忽略禁用的注释。

[0062] 图 7 是带有注释的一示例性本地化项目数据格式 700 的框图。在一个示例性实施方式中，本地化项目 702 包括属性 704。属性 704 包括项目类型 706 和项目 ID708。本地化项目 702 还包括串元素 710、二进制元素 712 和注释 714。本地化项目 702 是可被翻译或以其它方式适于本地文化和语言的正进行本地化的软件的任何部分或源。例如，文本消息或图标是本地化项目 702。项目类型 706 是指定本地化项目 702 的类型的属性。例如，项目类型 706 可指示特定本地化项目 702 是文本消息或色彩。项目 ID 708 用作本地化项目 702 的标识符。本地化项目 702 可任选地包括串 710、二进制数据 712 和注释 714，这取决于项目类型 706。例如，如果项目类型 706 指示本地化项目 702 是文本串，则本地化项目 702 可包括指定在本地化中使用的默认字体的另一元素，诸如特性元素（未在图中示出）。在一个示例性实施方式中，本地化项目 702 中包含若干类型的串 710 和二进制数据 712。例如，本地化项目 702 中可以包括用于源语言、目标语言和其它参考语言的串和二进制数据。参考语言可用于为本地化项目 702 从源语言向目标语言的翻译提供附加信息。在一个示例性实施方式中，父本地化项目 702 包括一起构成本地化项目 702 的分层结构的零个或更多其它子本地化项目 702（未示出）。子本地化项目 702 通过指针或等效软件技术的方式包括在父本地化项目 702 中。

[0063] 图 8 是一示例性自有注释数据格式 806 的示意图。所示示例性自有注释元素 802 包括多个注释元素 602，每个元素包括属性 804。属性 804 包括名称属性 806。

[0064] 图 9A-9D 示出设置元素 902 和相应 XML 元素的示例性实施方式。图 9A 是一示例性设置数据格式的示意图。设置元素 902 包括含有名称 906 的属性 904。设置元素 902 还包括多个设置 908 项目。示例性设置 1 包括属性 910。属性 910 包括设置 1 的名称 912。示例性设置 1 还包括布尔元素 914、整数元素 916、枚举元素 918、串元素 920、列表元素 922 和选项表元素 924。以下进一步描述设置 908 的每个元 素。设置元素 902 指定本地化数据文件的当前设置。

[0065] 图 9B 是一示例性枚举元素 942 的示意图。示例性枚举元素 942 包括含有名称 946 和值 948 的属性 944。名称属性 946 通过名称标识枚举元素 942。值属性 948 包括由枚举 942 表示的枚举的值。

[0066] 图 9C 是一示例性列表元素 962 的示意图。示例性列表元素 962 包括含有名称 966 和多个项目元素 968 的属性 964。名称属性 966 通过名称标识列表元素 962。项目元素 968 各自表示列表 962 的一个条目，并且可包括许多属性（未在本图中示出），诸如项目标识符、序列号、源文件名称等。此外，项目 968 可包括其它元素（未在本图中示出），诸如串元素、二进制元素、注释元素等。

[0067] 图 9D 是一示例性选项表元素 982 的示意图。示例性选项表元素 892 包括含有名称 986 和值属性 988 的属性 984。示例性选项表元素 982 还包括多个项目元素 990。名称属性 986 通过名称标识列表元素 982。项目元素 990 各自表示列表 982 的一个条目，并且可包括许多属性（未在本图中示出），诸如项目标识符、序列号、源文件名称等。此外，项目 990 可包括其它元素（未在本图中示出），诸如串元素、二进制元素、注释元素等。

[0068] 本领域技术人员将会意识到，数据元的其它变型也是可能的。例如，数据元可包括

诸如“任何 (Any)”元素（未在上图中示出）的一类数据元，其中任何 (Any) 元素包括名称属性、类型属性和值属性。在这种示例性实施方式中，类型属性指定应该如何解读值属性。例如，类型可以是“Unsigned_Integer（未赋值整数）”以及值可以是“15”。

[0069] 图 10 是一示例性本地化 XML 数据格式 1002 的示意图。本地化 XML 数据格式用于定义在本地化进程中使用的本地化数据的总的格式。示例性本地化 XML 数据元 1002 包括属性 1004 和任选元素，诸如如上所述的设置 1016、特性包 1018、自有注释 1020 和本地化项目 1022。属性 1004 包括名称属性 1006 以及其它任选属性，诸如解析器 ID 1010、描述 1012、源 1012 和目标 1014。如上参照图 7 所述，本地化项目 1022 可包括共同构成本地化项目 702 的分层结构的零个或更多子本地化项目 1022（未示出）。

[0070] 图 11A 是包含在 CDATA XML 元素 1102 中的文本数据的一示例性示意图。如本领域中已知的，CDATA XML 元素 1102 用于表示本地化数据文件中的自由文本 1104，类似于串。包含在 CDATA XML 元素 1102 中的自由文本 1104 通过在最后一个自由文本 1104 字符结束处立即使用闭合双括号 1106 来划定。即，在自由文本 1104 的结束处插入闭合双括号 1106，而在自由文本 1104 的最后一个字符与闭合双括号 1106 之间没有任何空字符，诸如空白、制表符等。图 11B 是包含在 CDATA XML 元素 1102a 中的含有方括号字符 1110 的文本数据的一示例性示意图。如果自由文本 1104a 包括方括号“]”字符 1110，则不能清晰确定自由文本 1104a 的范围。当字符的含义模糊时，即当能以多种方式解释字符时，转义字符可用于限制字符的解释。除其它技术之外可使用转义字符来消除该字符的岐义。在一个示例性实施方式中，消除方括号“]”的岐义包括在方括号字符 1110 之前插入诸如空格或制表符符号的空字符 1108，来标识方括号 1110 是自由文本 1104a 的一部分且不是闭合双括号 1106a 的一部分。在其它示例性实施方式中，可类似地使用其它特定字符。

[0071] 图 12 是本地化 XML 模式 1202 和相应对象模型 1208 的一示例性关系 1200 的框图。对象模型 1208 包括多个类 1206，每个类 1206 指定对象模型中的软件对象的设计。本领域技术人员将会意识到，类是用于以诸如 C++(C 加加）、C#(C 井号）和 Java 的面向对象计算机语言定义软件对象的抽象对象。此外，本领域技术人员将会认识到，软件对象是通过例示类而在计算机存储器中创建的，即通过分配存储器以基于由对应类指定的格式在存储器中创建物理对象。对象模型 1208 实质上对本地化 XML 模式 1202 中的每个元素 1204 定义一个类 1206。使用对象模型 1208，在软件应用程序和工具 1210 中实现本地化 XML 模式。如上所述，元素 1204 定义本地化数据格式 1212。用于如元素 1204 所原始定义的用于指定本地化数据的数据格式的数据接口 1214 由对象模型 1208 中的类 1206 指定。软件应用程序和工具 1210 使用数据接口 1214 来对每个数据片采用正确格式以正确访问和操纵本地化数据。软件应用程序和工具 1210 还使用功能接口 1216 来访问和操纵本地化数据以配置和执行本地化任务。

[0072] 图 13A 是带有返回指针的一示例性父和子对象的框图。父对象 1302 是从第一类例示的软件对象。子对象 1304 是从第二类例示的软件对象，该第二类是在设计第一和第二类时从第一类导出的。本领域技术人员将会认识到，在诸如 C++、C# 和 Java 的面向对象的计算机语言中，第二类可从第一类导出（即指定）。第二类被认为是继承在第一类中包含的成员。类的成员包括函数、变量、指针和其它类。第二类可定义在第一类中未定义的附加新成员。在本领域中，通过第二类获得第一类的成员的关系称为继承。继承是通常在与运

行时（即在软件执行过程中）不同的软件开发过程中的设计时发生的进程。本领域技术人员公知为包容的面向对象语言的另一特性也称为聚集。当第一类是第二类的成员时，第一类被认为是包含在第二类 中。包容是与继承不同的对象间关系。本领域中，习惯在继承和包容关系中使用术语“父”表示第一类，术语“子”表示第二类。因此，以下讨论中将使用父/子的术语。在图 13A 所示的示例中，对象模型 1208（图 12）中的子对象 1304 包括指向对应父对象 1302 的返回指针 1306。返回指针 1306 通过在父对象 1302 与子对象 1304 之间提供直接链接来增加系统性能，由此可以遍历对象模型 1208 中的对象关系。所有对象使用指针 1306 维护对其相应父对象的引用。当父对象 1302 与子对象 1304 之间建立关系时，由父对象 1302 设定反向指针 1306。当父对象 1302 与子对象 1304 之间的关系断开时，将子对象 1302 设定为指向另一父对象。

[0073] 图 13B 是带有指针和文件指针的一示例性父对象和子对象的框图。如上所述，当父对象 1322 与 1324 之间建立关系时，父对象 1322 设定返回指针 1326。当父对象 1322 与子对象 1324 之间的关系断开时，将子对象 1322 设定为指向另一父对象。当子对象 1324 是资源对象时，文件指针 1328 由子对象 1324 用来指向资源文件。如本领域技术人员已知的，资源通常是表示诸如图标、菜单或位图的图像组件的图形数据对象。资源数据包含在根据已起草的资源规范使用资源编译器创建的资源文件 1330 中。

[0074] 图 14 是带有外部定制文化信息的一示例性对象模型 1402 的示意图。对象模型 1402 包括默认的公知定制文化 1404。为了本地化到对象模型 1402 中未默认包含的语言和文化，用来自外部文件 1406 的定制文化信息 1408 增添对象模型 1402。在一个示例性实施方式中，外部文件 1406 呈现在本地系统中。在另一示例性实施方式中，外部文件 1406 位于远程系统上。在一示例性实施方式中，对定制文化信息 1406 进行手动更新。在另一示例性实施方式中，可通过本地化应用程序软件将定制文化信息 1406 写入文件 1406。

[0075] 图 15 是部分加载数据的一示例性方法 1500 的功能流程图。在框 1510，客户端软件应用程序打开本地化数据文件。如上所述，本地化数据文件包括在本地化进程中由也称为客户端软件应用程序的软件应用程序工具使用的本地化数据。在一个实施方式中，本地化数据文件包括 XML 元素。本领域技术人员将会意识到，可使用其它方法和格式来表示软件应用程序的数据，因此，本文中有关示例性 XML 元素的讨论被解读为是示例性而非限制性的。客户端软件应用程序具有确定将哪些 XML 元素加载到存储器中以供处理的内部逻辑。例如，只处理用于本地化的文本信息的客户端软件应用程序只需要加载文本相关信息，诸如文本字符的字体和大小。在框 1520，从本地化数据文件获取下一 XML 元素以加载到存储器中并进行处 理。然后，在框 1530，客户端软件应用程序判定是否必须加载当前 XML 元素。如果当前 XML 元素被选择加载，则在框 1540，加载当前 XML 元素，且方法 1500 进行到框 1550。如果当前 XML 元素未被选择加载，则方法 1500 进行到框 1550，其中方法 1500 判定在本地化数据文件中是否有更多 XML 元素可用。如果在本地化数据文件中有更多 XML 元素可用，则方法 1500 返回到框 1520 以获取下一 XML 元素。如果在本地化文件中没有更多的 XML 元素可用，则方法 1500 结束。在一个示例性实施方式中，客户端软件应用程序选择加载一种通用类型数据而不加载其它通用类型数据。例如，处理文本的客户端设置标志，用于只加载串数据而不加载任何二进制数据。在这种情形中，数据的选择在总体级别上进行，从而区分数据类型以基于所选的一般类型数据来加载。

[0076] 图 16 是用回调功能细化数据加载的一示例性方法 1600 的功能流程图。在一个实施方式中，客户端软件应用程序指定要以细化级别加载的数据，包括所有类型的数据，诸如串数据和二进制数据。与以上参照图 15 讨论的在数据类型总体级别上操作的部分数据加载不同，细化数据加载在所有数据类型内的精细级别上进行。在细化加载中，客户端软件应用程序根据加载哪个数据元提供具体标准。在一个示例性实施方式中，客户端软件应用程序向对象模型 1208 的功能接口提供回调功能，由此从本地化数据文件检索本地化数据的对象判定是否加载每个数据元。回调功能使用如客户端软件应用程序所定义的用于选择数据的标准。在框 1610 中，客户端软件应用程序打开本地化数据文件。方法 1600 进行到框 1620，其中由客户端软件应用程序从访问本地化数据文件的对象模型 1208 向对象提供回调功能。在框 1630，获取 XML 数据元。该对象使用由客户端软件应用程序提供的回调功能以针对加载对 XML 数据元进行评估。在框 1650，方法 1600 基于来自回调功能的结果判定是否加载当前 XML 数据元。如果当前 XML 数据元被选择用于加载，则方法 1600 进行到框 1660，其中 XML 数据元被加载到存储器中，且方法进行到框 1670。如果当前 XML 未被选择用于加载，则方法 1600 进行到框 1670。在框 1670，方法 1600 判定在本地化数据文件中是否有更多的 XML 数据元可用。如果有更多的 XML 数据元可用，则方法 1600 返回框 1630 以获取下一 XML 数据元。否则，方法 1600 结束。

[0077] 图 17A 是数据部分保存的一示例性方法 1700 的功能流程图。部分保存方法 1700 是部分加载方法 1500 的补充。可以要求在存储器中具有准备保存到数据文件中的本地化数据的客户端软件应用程序只保存该数据的一部分。部分保存方法 1700 允许客户端软件应用程序指定应该将哪些数据保存到数据文件。例如，可以要求客户端软件应用程序只保存串数据。客户端软件应用程序可指定只将串数据保存到数据文件中。在框 1710，客户端软件应用程序指定要保存到数据文件中的数据类型。在框 1720，将由客户端软件应用程序所指定类型的数据保存到数据文件中。在框 1730，关闭数据文件，方法 1700 结束。

[0078] 图 17B 是合并数据保存的一示例性方法 1750 的功能流程图。无论通过上述的部分数据加载还是细化数据加载，客户端软件应用程序只在存储器中加载数据的一部分。如果客户端软件应用程序的存储器中的数据被如此存储，则最初未加载到存储器中的所有数据将丢失并且未记录在输出数据文件中。为了防止数据丢失，方法 1750 将客户端软件应用程序存储器中的数据与来自最初未加载到客户端软件应用程序存储器中的数据文件合并。为了保存由客户端软件应用程序对已加载数据进行的修改，保存对数据文件和客户端软件应用程序存储器通用的该数据的存储器副本。在框 1760，打开客户端软件应用程序加载本地化数据所使用的原始数据文件。在框 1770，从该数据文件获取下一可用 XML 元素。在框 1780，方法 1750 判定客户端软件应用程序的存储器中是否也存在从该数据文件获取的当前 XML 元素。如果存储器中存在当前 XML 元素，则在框 1785，将 XML 元素的存储器副本保存到该数据文件中。然后，方法 1750 进行到框 1790。在框 1780，如果存储器中不存在当前 XML 元素，则当前 XML 元素最初未被加载，并且未被修改，因此无需再次在数据文件中保存。在这种情形中，方法 1750 进行到框 1790。在框 1790，方法 1750 判定数据文件中是否还有更多的 XML 元素。如果还有更多的 XML 元素，则方法 1750 进行到框 1770 以获取下一 XML 元素。否则，方法 1750 结束。

[0079] 图 18 是细化数据保存的一示例性方法 1800 的功能流程图。细化保存方法 1800

是细化数据加载方法 1600 的补充。类似于细化加载方法 1600，细化保存方法 1800 指定每个数据对象是否必须被保存。在一个实施方式中，客户端软件应用程序指定要在精细细化级别保存的数据，包括所有类型的数据，诸如串数据和二进制数据。与以上参照图 17A 所述的在数据类型的总体级别上操作的数据部分保存不同，数据的细化保存在所有数据类型内的精细级别上进行。在细化保存中，客户端软件应用程序根据保存哪个数据对象指定具体标准。在一个示例性实施方式中，客户端软件应用程序向对象模型 1208 的功能接口提供回调功能，由此将本地化数据保存到本地化数据文件的对象判定是否保存每个数据元。回调功能使用如客户端软件应用程序所定义的数据选择标准。方法 1800 进行到框 1810，其中由客户端软件应用程 序从访问本地化数据文件的对象模型 1208 向对象提供回调功能。在框 1820，从客户端软件应用程序存储器获取 XML 数据元。在框 1830，该对象使用由客户端软件应用程序提供的回调功能来针对保存对 XML 数据对象进行评估。在框 1840，方法 1800 基于来自回调功能的结果判定是否保存当前 XML 数据元。如果当前 XML 数据元被选择为保存，则方法 1800 进行到框 1850，其中 XML 数据元被保存到数据文件，且该方法进行到框 1860。如果当前 XML 未被选择为保存，则方法 1800 进行到框 1860。在框 1860，方法 1800 判定在存储器中是否有更多的 XML 数据元可用。如果有更多的 XML 数据元可用，则方法 1800 返回框 1820 以获取下一 XML 数据元。否则，方法 1800 结束。

[0080] 图 19 是一示例性本地化对象模型 1900 的示意图。如上所述，本领域技术人员应该认识到，对象模型是软件系统中不同对象类型或类（对象的抽象表示）的关系的抽象表示。对象模型可用于表示对象之间的继承关系以及包容关系。对象模型 1900 提供用于对象和关系的类型规范，该规范允许相对于基于本地化 XML 模式 1202 创建的本地化数据文件中数据的基本输入和输出功能。如上所述，本领域技术人员将会意识到对象模型 1900 可应用于其它类型的数据模式，并且示例性本地化 XML 模式的讨论不应被解读为对本发明的限制。对象模型 1900 允许本地化数据文件的分割和合并。此外，对象模型 1900 允许添加有关注释的源和描述的信息。对象模型 1900 还允许包括已引用的翻译以在本地化进程中提供帮助。如参照图 12 所述的，对象模型 1900 紧密对应于本地化 XML 模式 1202。即，对象模型 1900 中的每个类对应于本地化 XML 模式 1202 中的一个元素。因此，本地化文件 1902 是表示基于本地化 XML 模式 1202 的本地化数据文件的类。本地化文件类 1902 包括文化类 1904 和本地化项目列表类 1906。本地化项目列表类 1906 包括在本地化项目类 1908 中。在一个实施方式中，本地化项目列表类 1906 是线性列表，与分层结构不同，它很容易允许将本地化数据文件 404 分割成部分数据文件 406 并将部分数据文件 406 合并回本地数据文件 408。本地化项目类 1908 是对象模型 1900 中对象模型 1900 的大多数其它类所相关的中心类。在一个实施方式中，本地化项目类 1908 包括父资源、本地化文件、资源 ID、子本地化项目（如下描述）的本地化项目列表（如下描述）、二进制数据类（如下描述）、以及各注释的注释列表（如下描述）。

[0081] 本地化项目类 1908 还包括注释类 1910。在一个实施方式中，注释类 1910 是注释列表类的一部分。本地化项目类 1908 还包括串数据类 1912 和二进制数据类 1914。串数据类 1912 包括串源类 1916 和串目标类 1918。串源数据类 1916 提供原始串和其它串特性。串目标类 1918 包括该串的本地化信息。二进制数据类 1914 包括二进制源类 1920 和二进制目标 1922。二进制源类 1920 呈现原始二进制字节数组和其它二进制特性。二进制目标

类 1922 提供二进制状态信息。在另一实施方式中，对象模型 1900 可以包括在本地化项目类 1908 中包含的诸如显示信息类和资源 ID 类的其它类。在一示例性实施方式中，串数据类 1912 和二进制数据类 1914 包括串参考类和二进制参考类（未示出）。参考类提供有关参考语言的信息，可用于为将串和二进制数据从源语言翻译到目标语言提供附加信息。

[0082] 图 20 是用于创建和提供本地化数据文件的一示例性方法 2000 的功能流程图。在框 2010，创建新的本地化数据文件。在框 2020，向本地化数据文件添加本地化项目。在框 2040，方法 2000 判定是否还有要向本地化数据文件添加的更多本地化项目。如果还有更多本地化项目，则方法 2000 进行到框 2020，其中向本地化数据文件添加本地化项目。否则，方法 2000 进行到框 2060，其中保存本地化数据文件。

[0083] 图 21 是用于向本地化项目添加注释的一示例性方法 2100 的功能流程图。方法 2100 需要来自客户端软件应用程序的输入信息，以标识添加注释所针对的本地化项目。如上所述，可在注释中嵌入与本地化项目相关联的用于本地化的指令。在框 2110，打开本地化数据文件。在框 2120，方法 2100 验证尝试向本地化项目添加注释的客户端软件应用程序的注释所有权。如果该客户端软件应用程序不拥有向本地化项目的注释类型，则方法 2100 进行到框 2170，其中关闭本地化数据。如果客户端软件应用程序拥有该注释类型，则方法 2100 进行到框 2130，其中创建新的注释。在框 2140，对所需值设定注释的名称和值属性。在框 2150，向本地化项目添加注释。方法 2100 进行到框 2160，其中保存该文件。在框 2170，关闭该文件，且方法 2100 结束。

[0084] 图 22 是用于文件剥离的一示例性方法 2200 的功能流程图。在保存数据之前，将必须不被保存到本地化数据文件的所有数据从客户端软件应用程序的存储器移除。在一个实施方式中，方法 2200 移除不包含注释的所有本地化项目。将二进制信息和串从所有其它本地化项目移除。在一个实施方式中，使用递归方法，它包括子本地化项目作为输入。递归方法从子本地化项目剥离所有二进制和串信息。来自对递归方法的调用的“假”返回值表明作为输入向递归方法提供的子本地化项目以及该子本地化项目的所有子项目不具有注释，并将该子本地化项目和该子本地化项目的所有子项目移除。在框 2210，访问子本地化项目。在框 2220，方法 2200 判定 该子项目是否包括注释。如果该子项目包括注释，则方法 2200 进行到框 2230，其中丢弃子本地化项目的二进制和串数据。如果该子项目不具有注释，则在框 2240 移除该子项目。在框 2230，方法 2200 进行到框 2250。在框 2240，该方法 2200 进行到框 2250，其中判定是否还有更多子本地化项目。如果还有更多子本地化项目，则方法 2200 进行到框 2210，其中访问下一子本地化项目以进行评估。否则方法 2200 结束。

[0085] 虽然示出和描述了本发明的当前较佳实施方式，但是本领域技术人员应该意识到，在此可进行各种变化而不背离本发明的精神和方法。例如，虽然上述系统和方法针对使用 XML 模式的本地化数据，但是可以使用其它数据格式规范。因此，本发明不应被解读为限制于上述示例性实施方式。

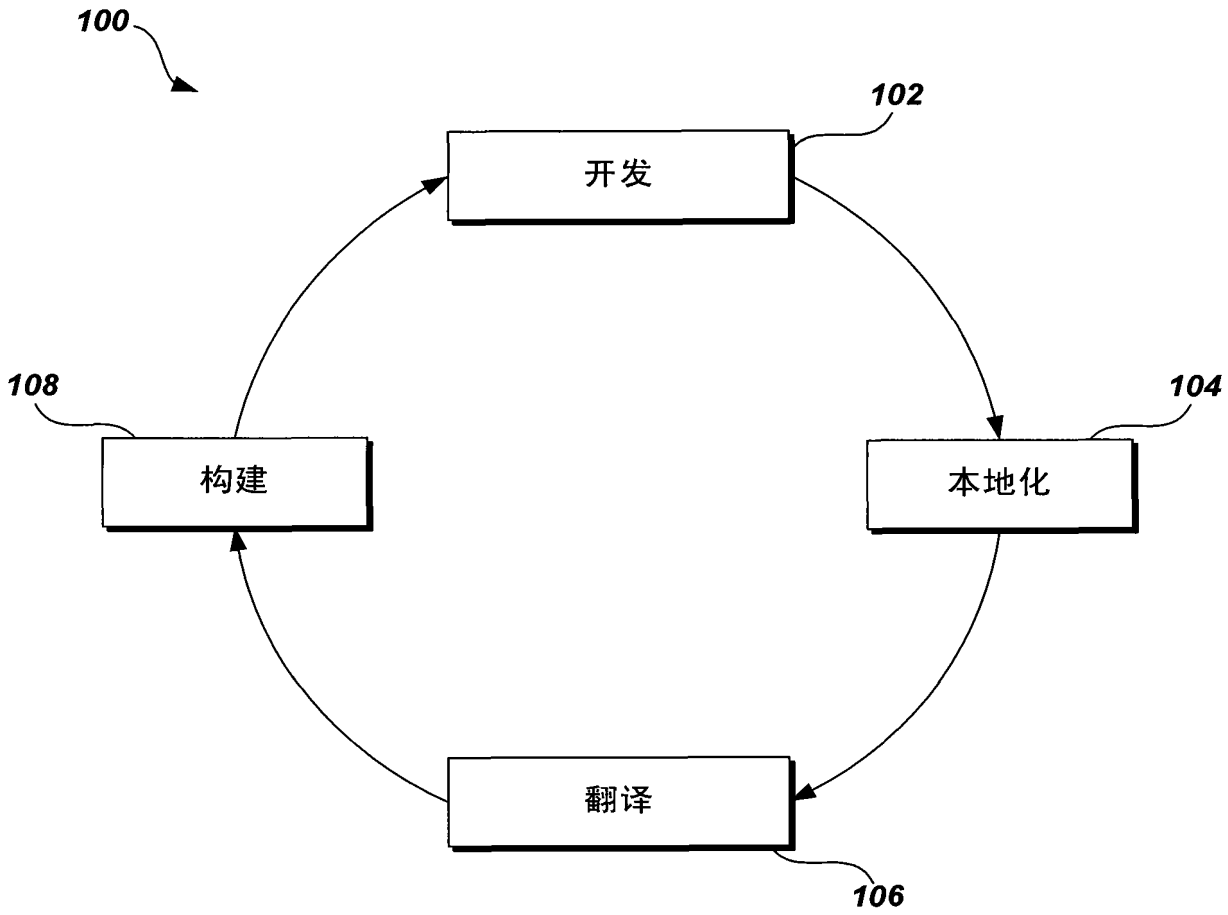


图 1

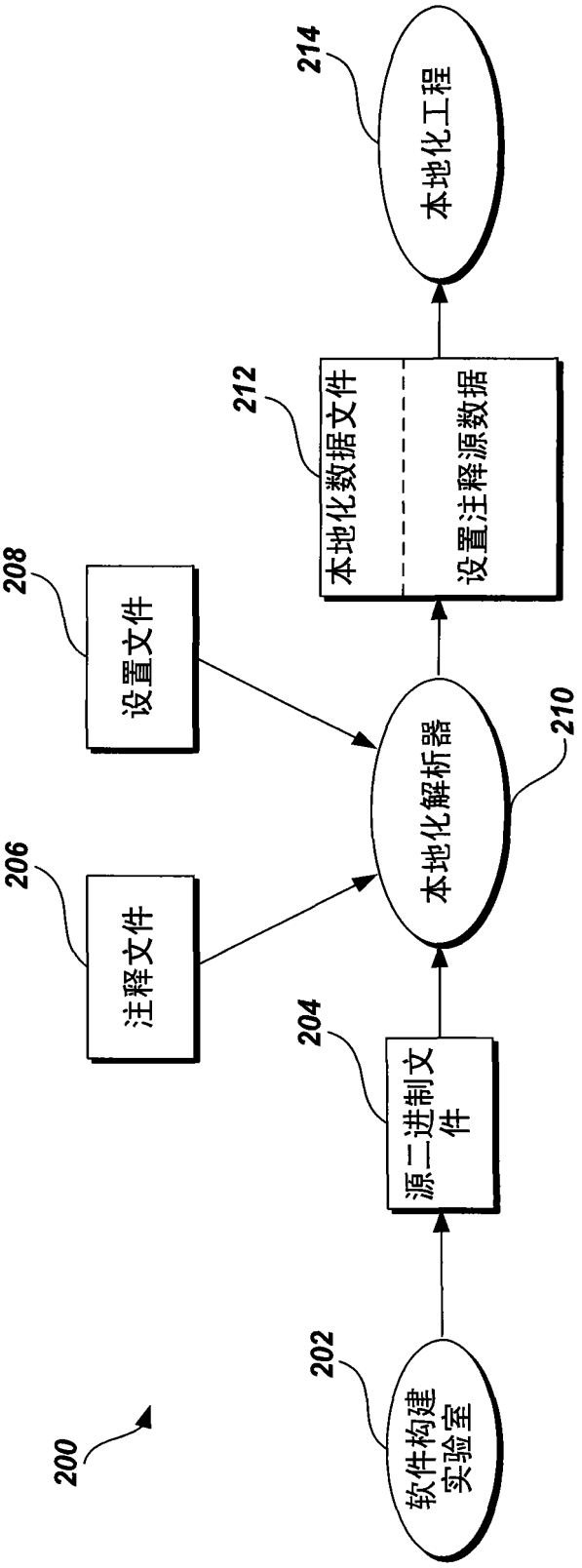

200
软件构建实验室
202
源二进制文件
204
注释文件
206
设置文件
208
本地化解析器
210
本地化数据文件
设置注释源数据
212
本地化工程
214

图 2

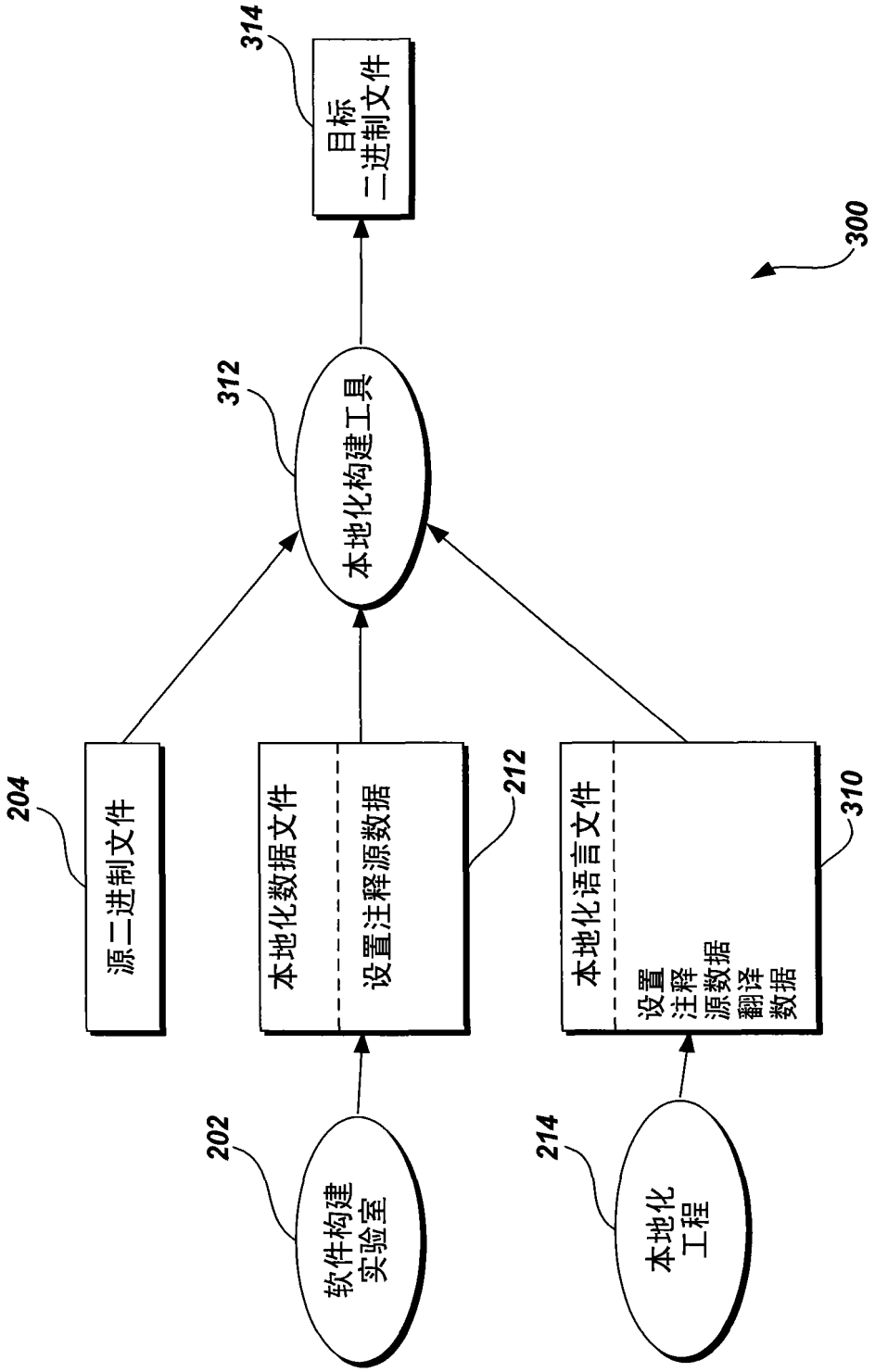

314
目标
二进制文件
300
312
本地化构建工具
204
源二进制文件
本地化数据文件
设置注释源数据
212
本地化语言文件
设置
注释
源数据
翻译
数据
310
202
软件构建
实验室
214
本地化
工程

图 3

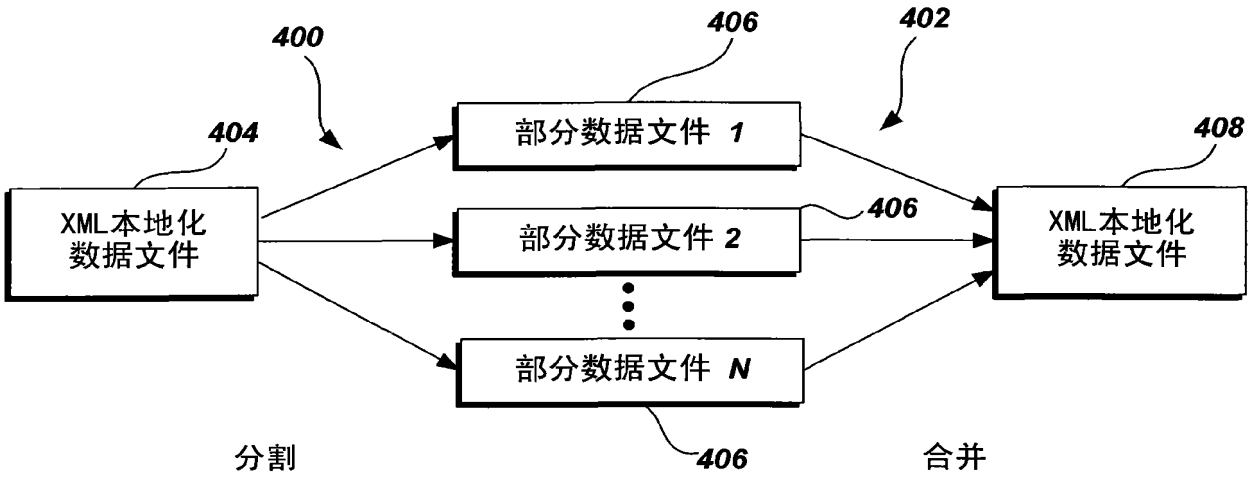

400
406
402
404
408
部分数据文件 1
XML本地化
数据文件
406
部分数据文件 2
XML本地化
数据文件
部分数据文件 N
406
分割
合并

图 4

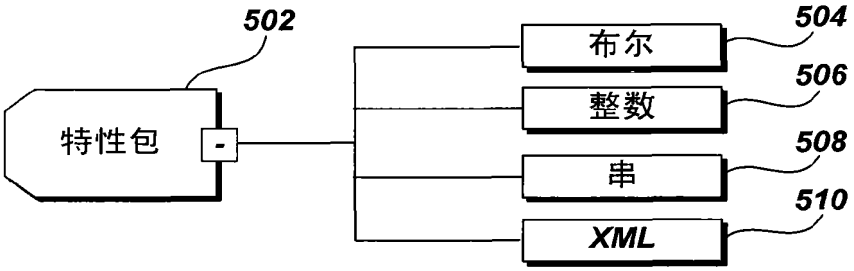


图 5A

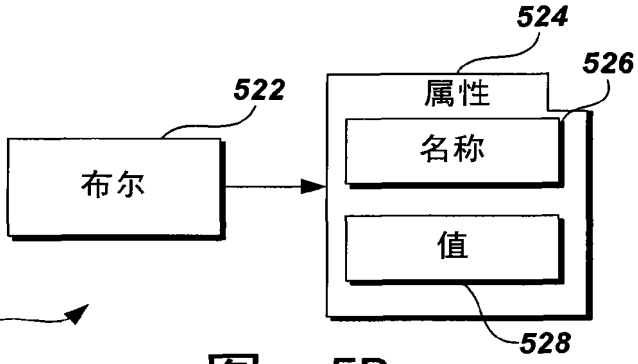


图 5B

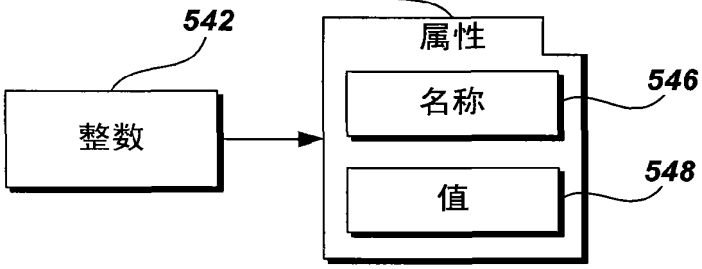


图 5C

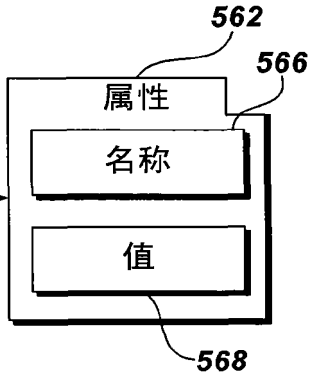


图 5D

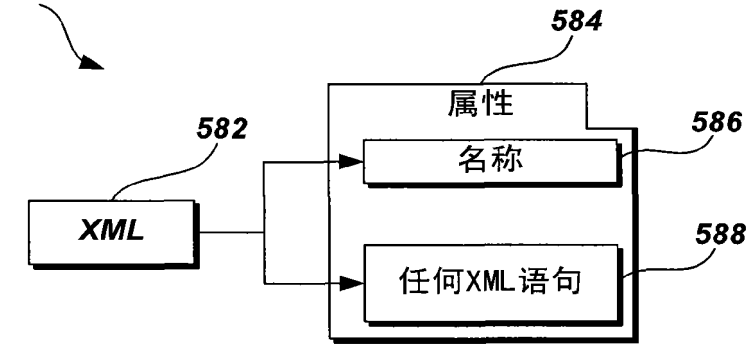


图 5E

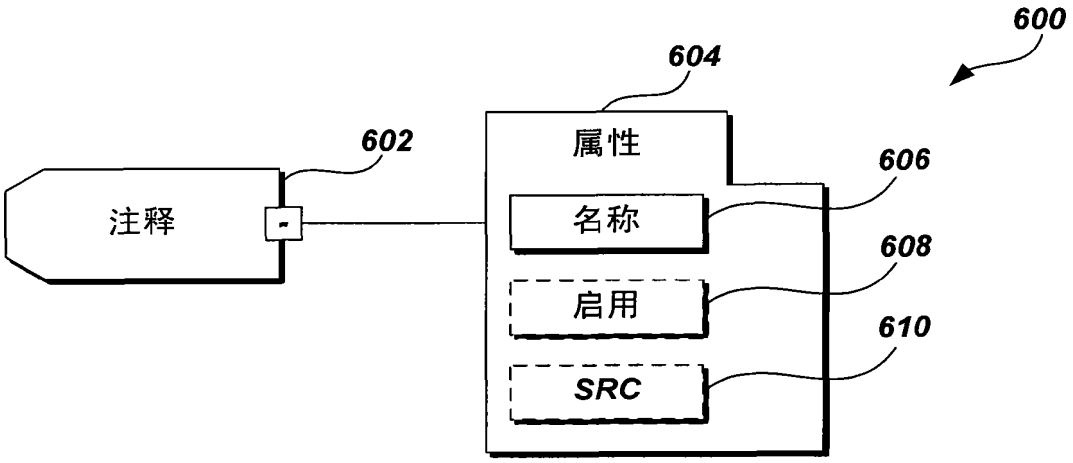

600
604
属性
602
注释
606
名称
608
启用
610
SRC

图 6

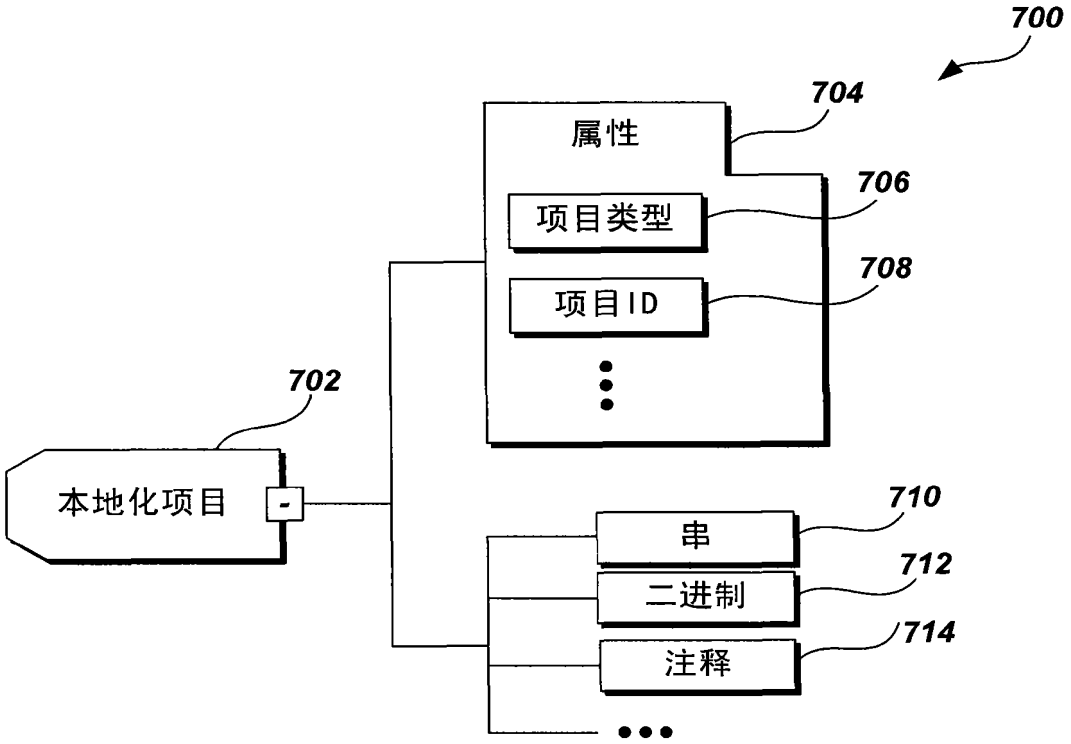

700
704
属性
706
项目类型
708
项目ID
702
本地化项目
710
串
712
二进制
714
注释

图 7

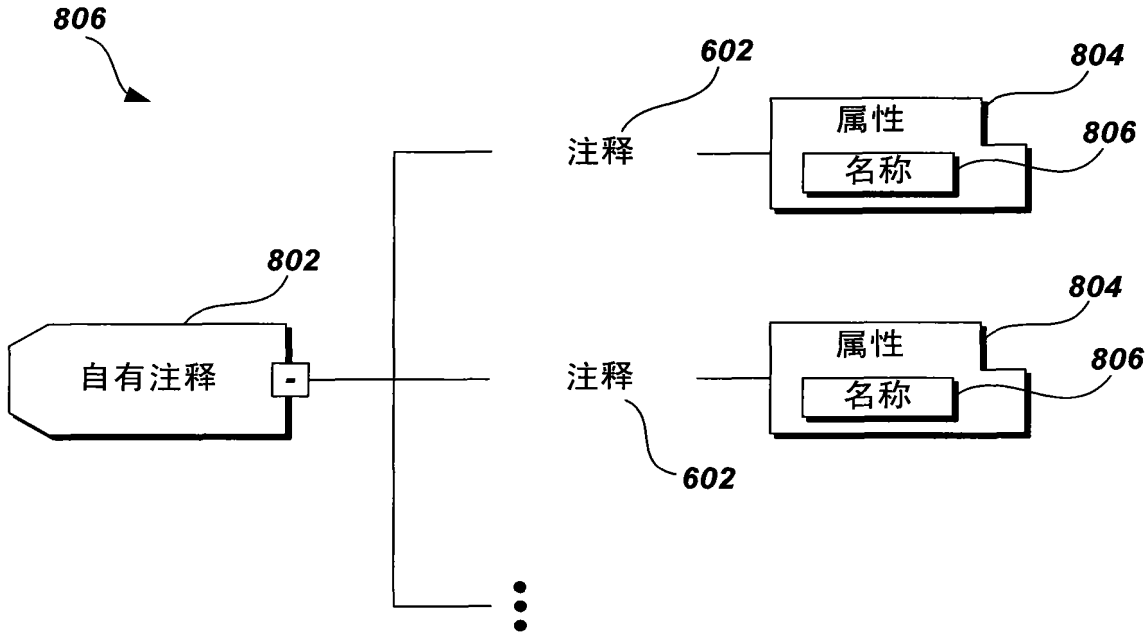


图 8

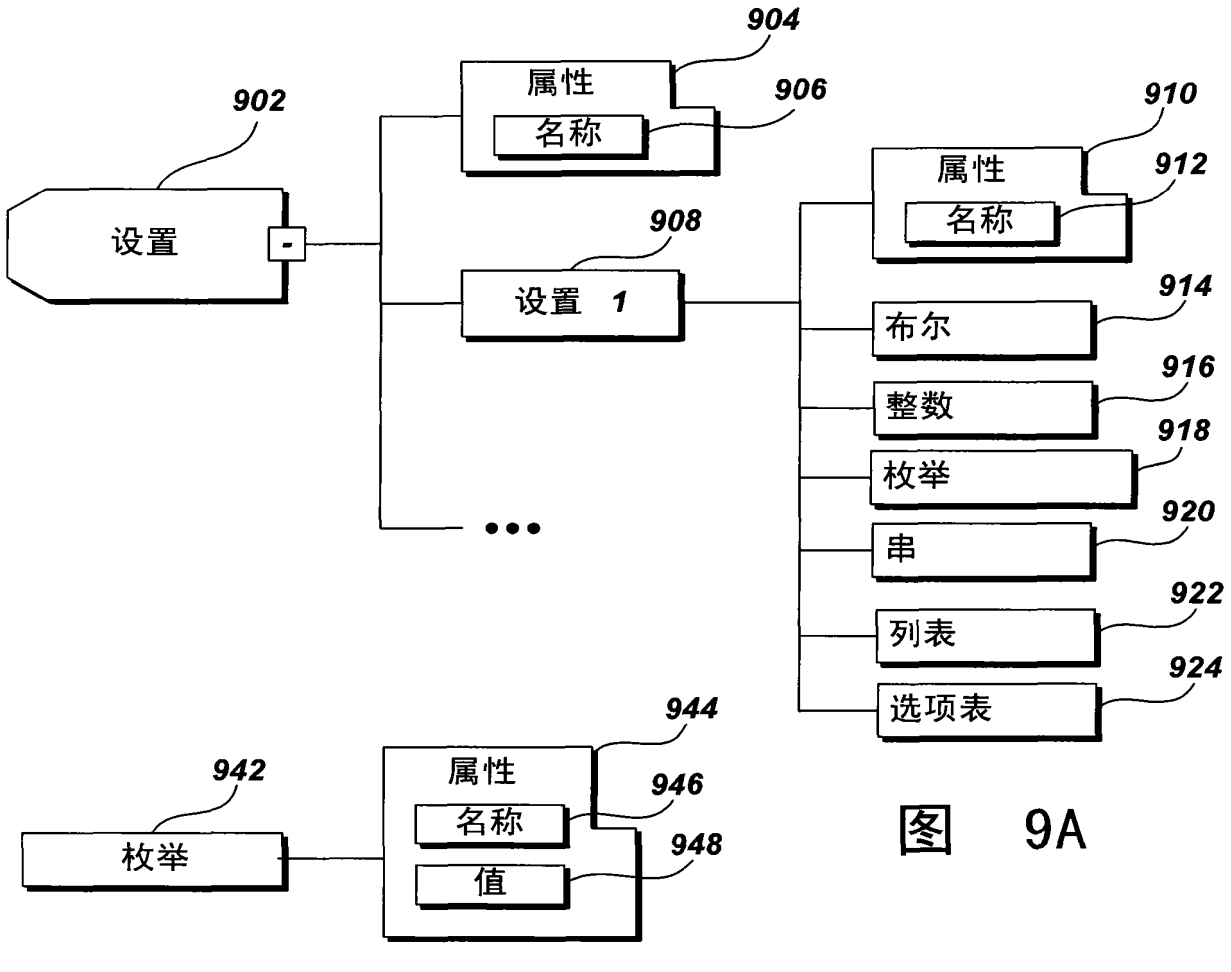


图 9A

图 9B

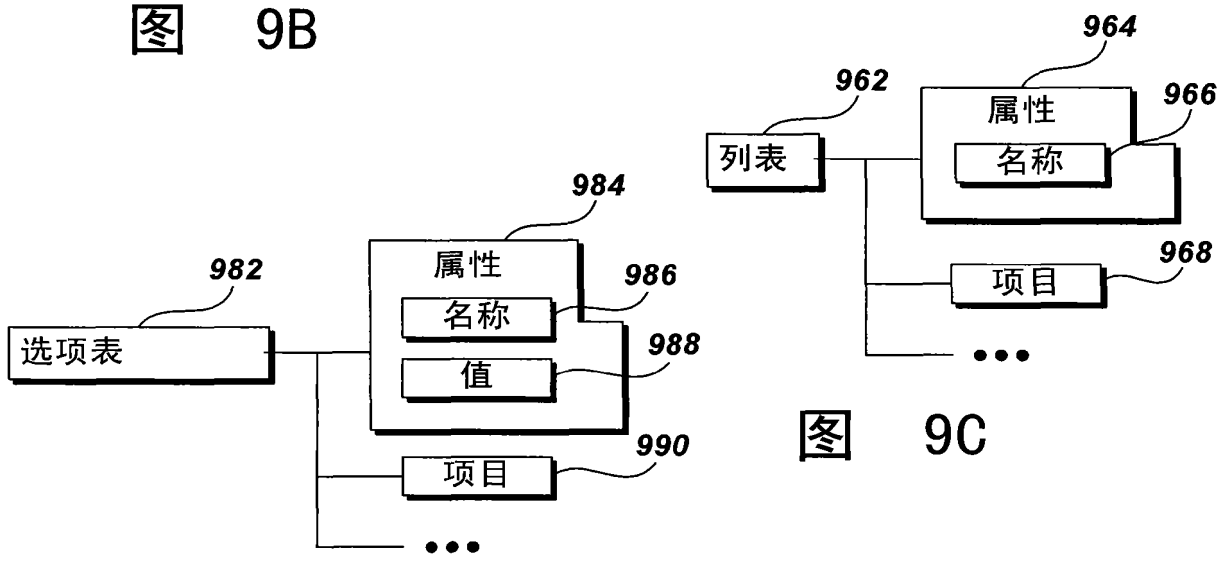


图 9C

图 9D

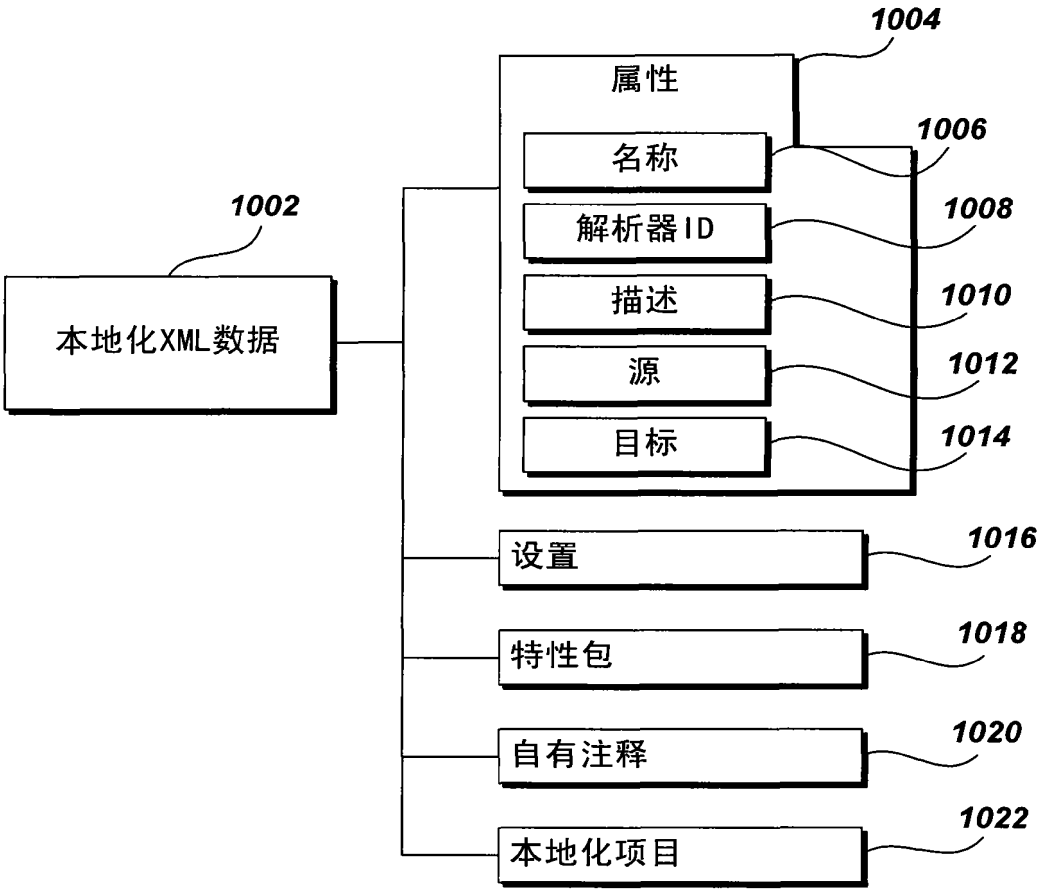


图 10

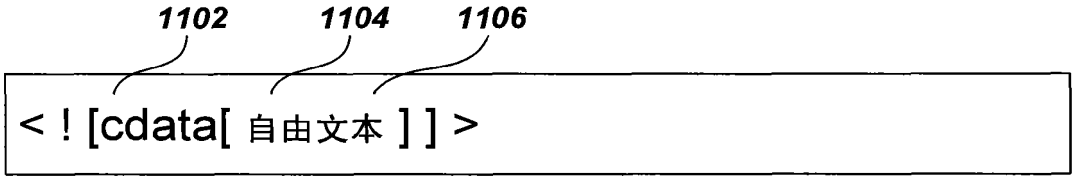


图 11A

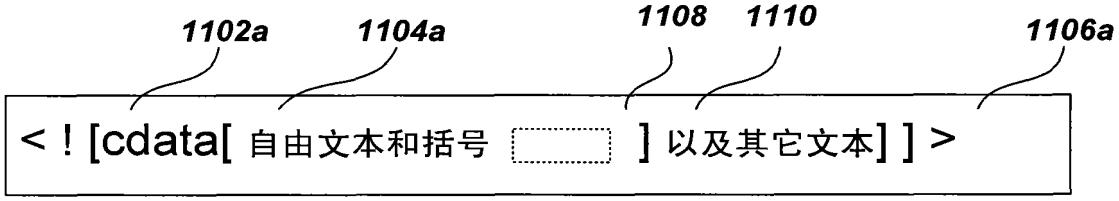


图 11B

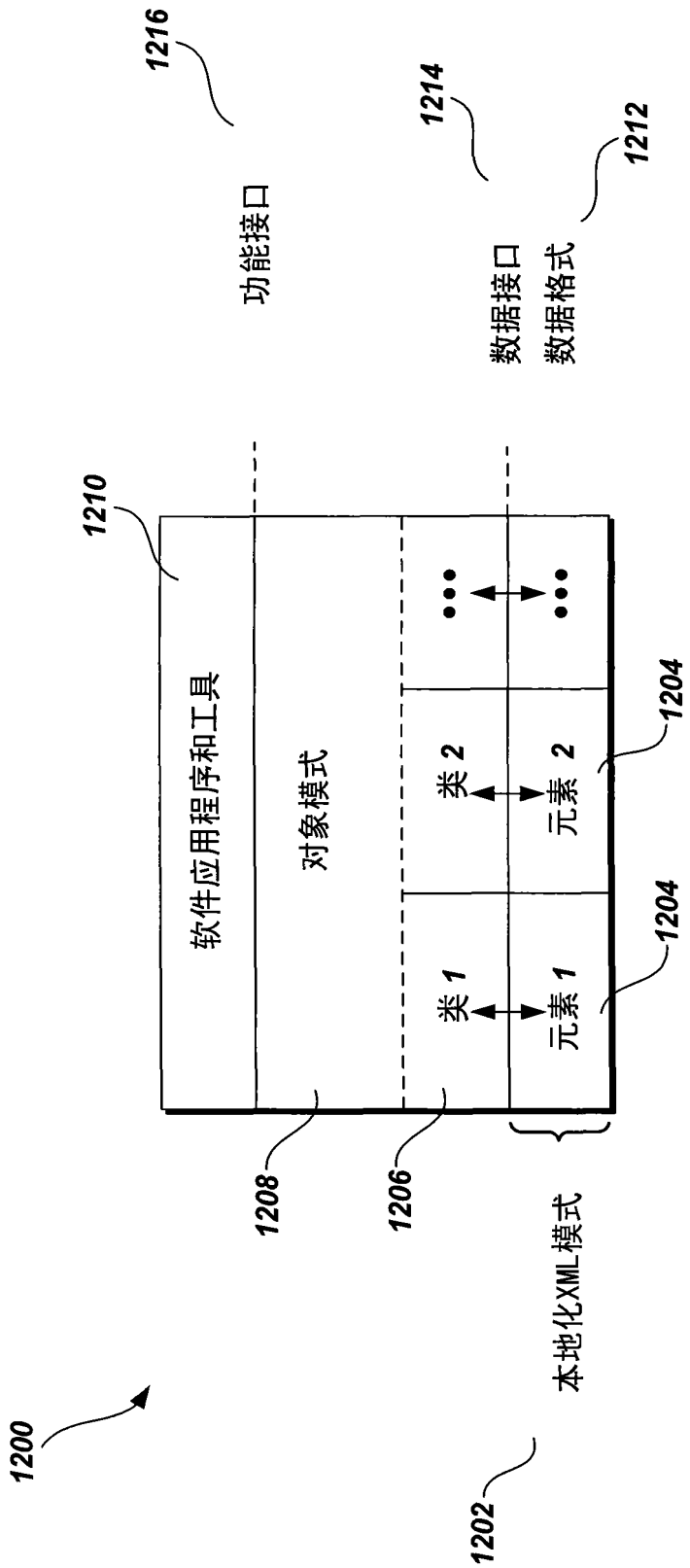

1216
功能接口
1214
数据接口
1212
数据格式
1210
软件应用程序和工具
对象模式
类 1
元素 1
类 2
元素 2
1204
1204
1208
1206
本地化XML模式
1200
1202

图 12

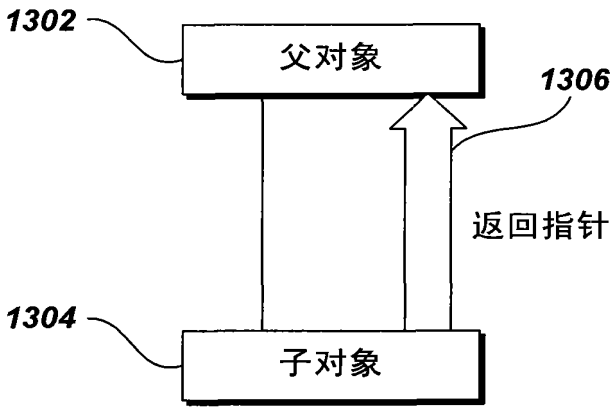

1302
父对象
1306
返回指针
1304
子对象

图 13A

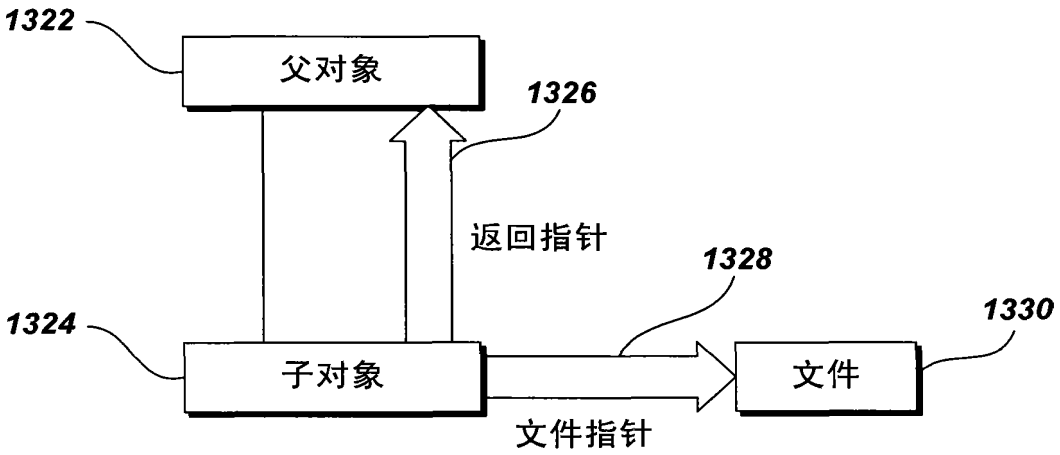

1322
父对象
1326
返回指针
1328
1330
1324
子对象
文件
文件指针

图 13B

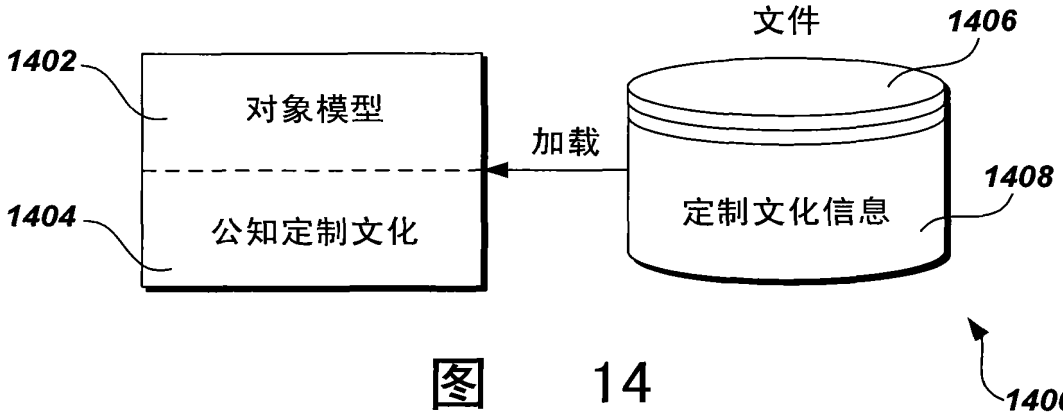

文件
1406
1402
对象模型
加载
1408
1404
公知定制文化
定制文化信息
1400

图 14

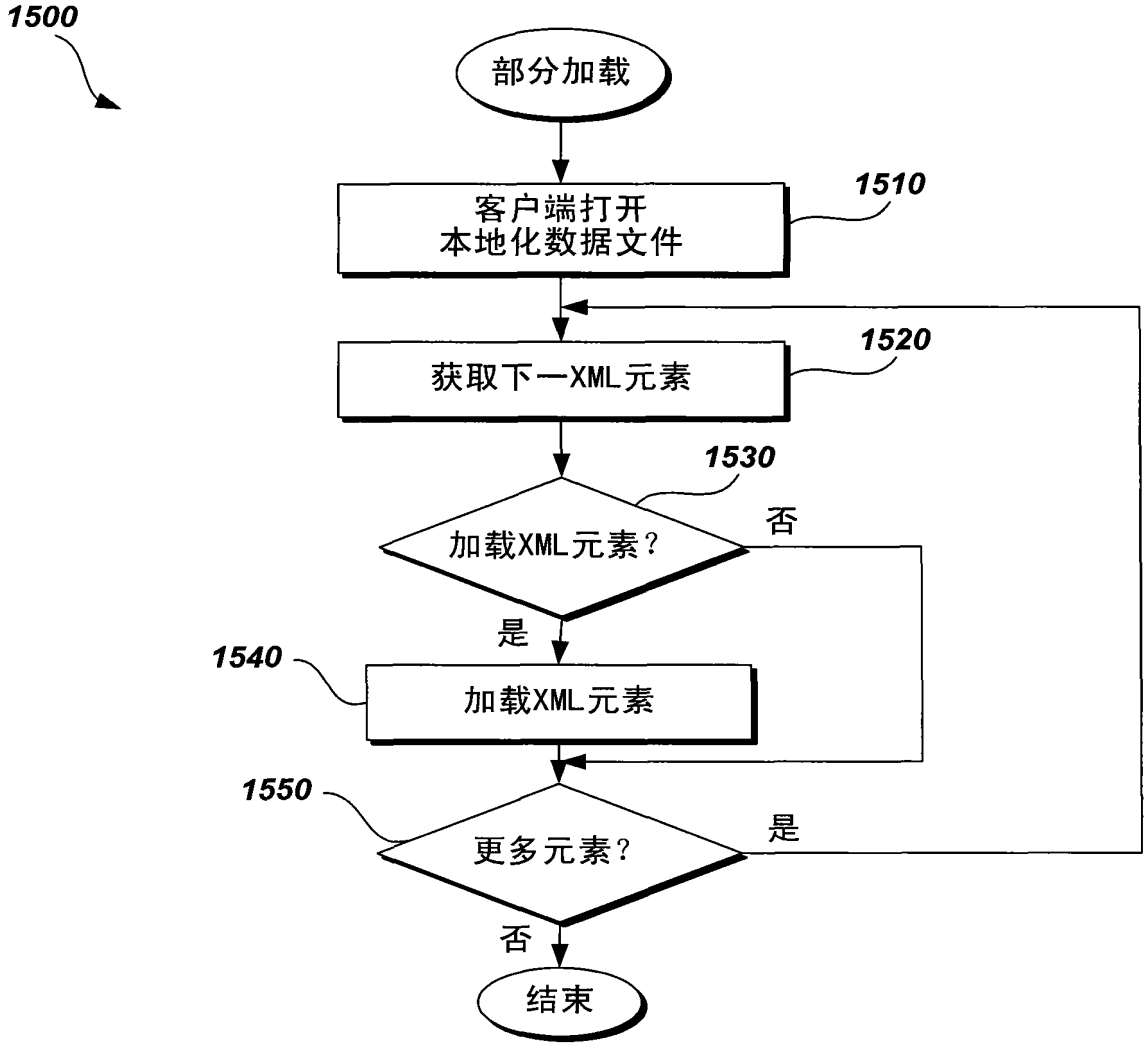

1500
部分加载
客户端打开
本地化数据文件
1510
获取下一XML元素
1520
1530
加载XML元素？
否
是
1540
加载XML元素
1550
更多元素？
是
否
结束

图 15

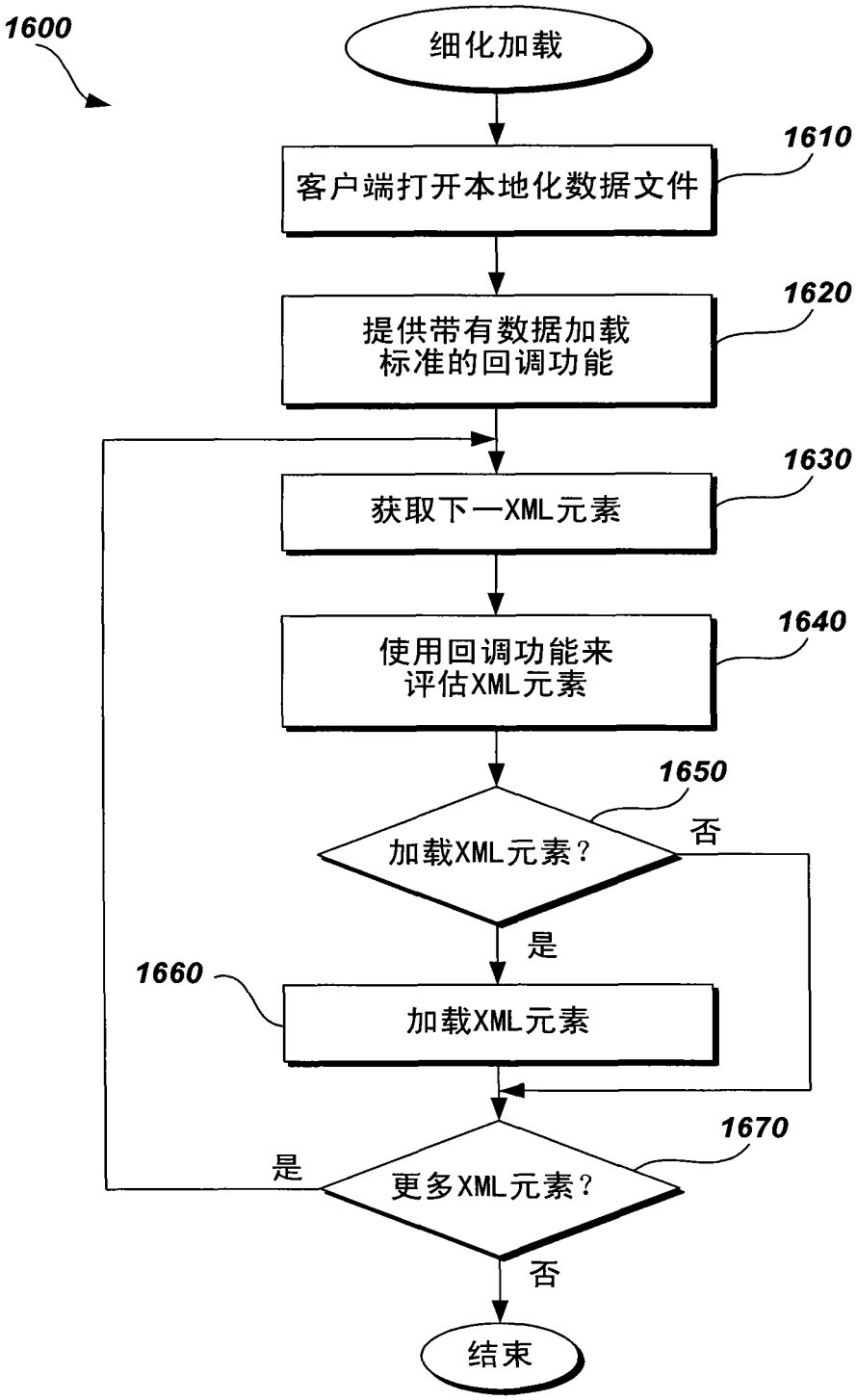

1600
细化加载
1610
客户端打开本地化数据文件
1620
提供带有数据加载
标准的回调功能
1630
获取下一XML元素
1640
使用回调功能来
评估XML元素
1650
加载XML元素？
否
是
1660
加载XML元素
1670
是
更多XML元素？
否
结束

图 16

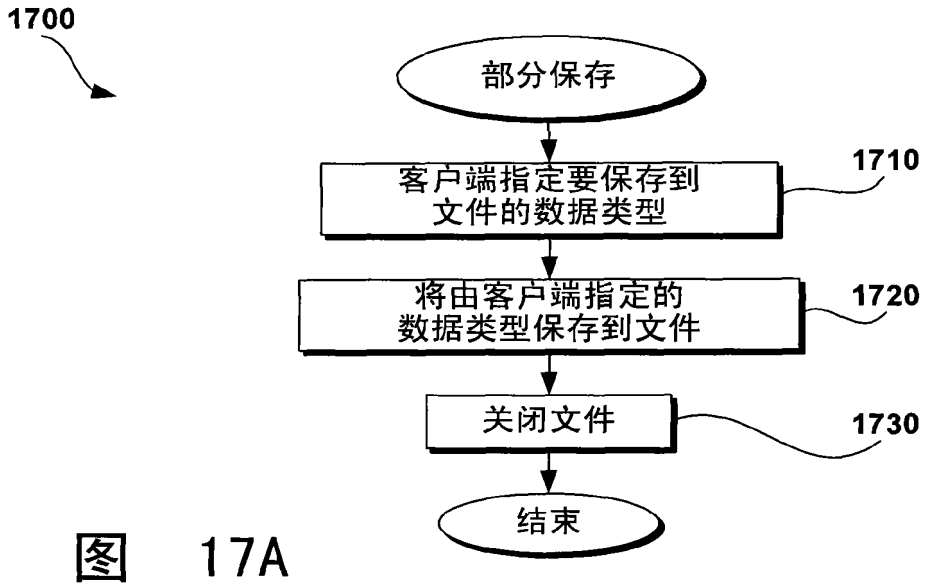

1700
部分保存
客户端指定要保存到文件的数据类型
1710
将由客户端指定的数据类型保存到文件
1720
关闭文件
1730
结束

图 17A

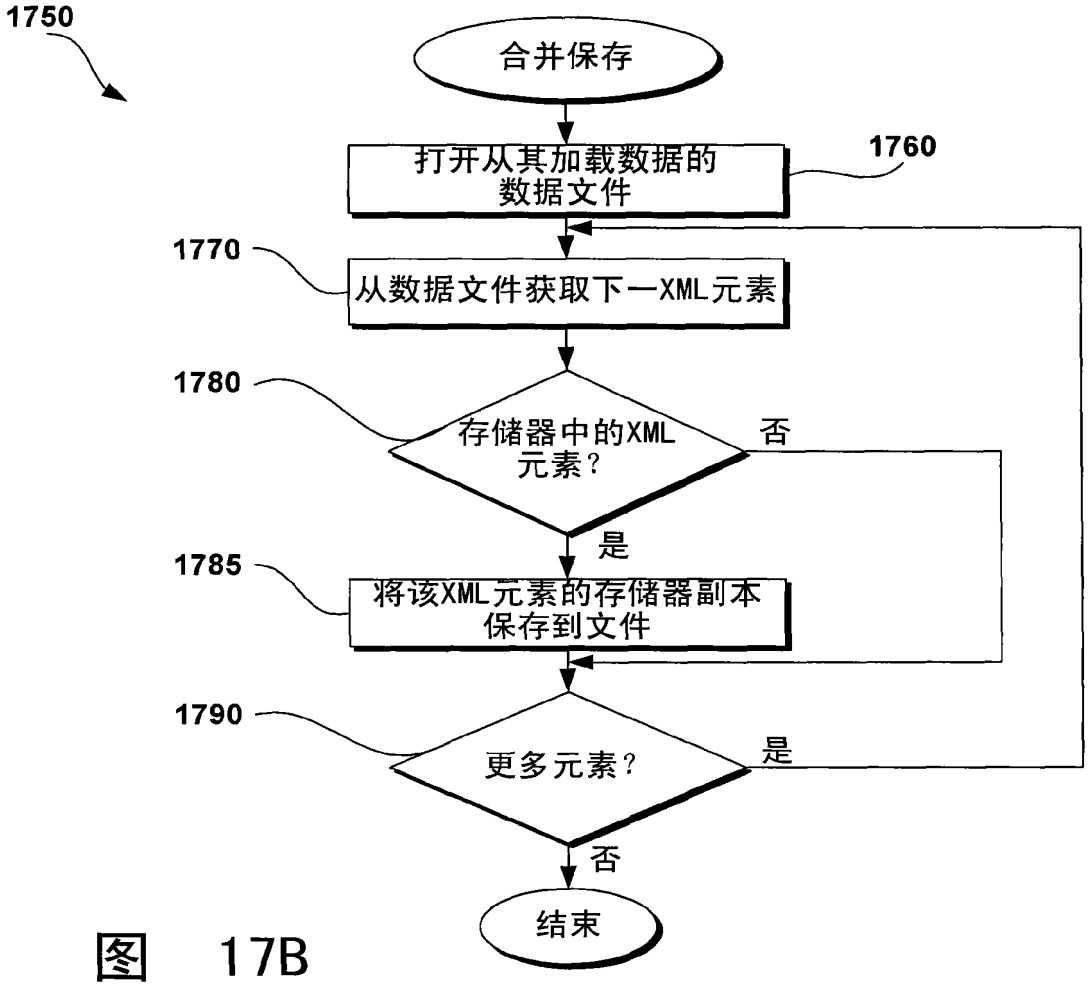

1750
合并保存
打开从其加载数据的数据文件
1760
1770
从数据文件获取下一XML元素
1780
存储器中的XML元素？
否
是
1785
将该XML元素的存储器副本保存到文件
1790
更多元素？
是
否
结束

图 17B

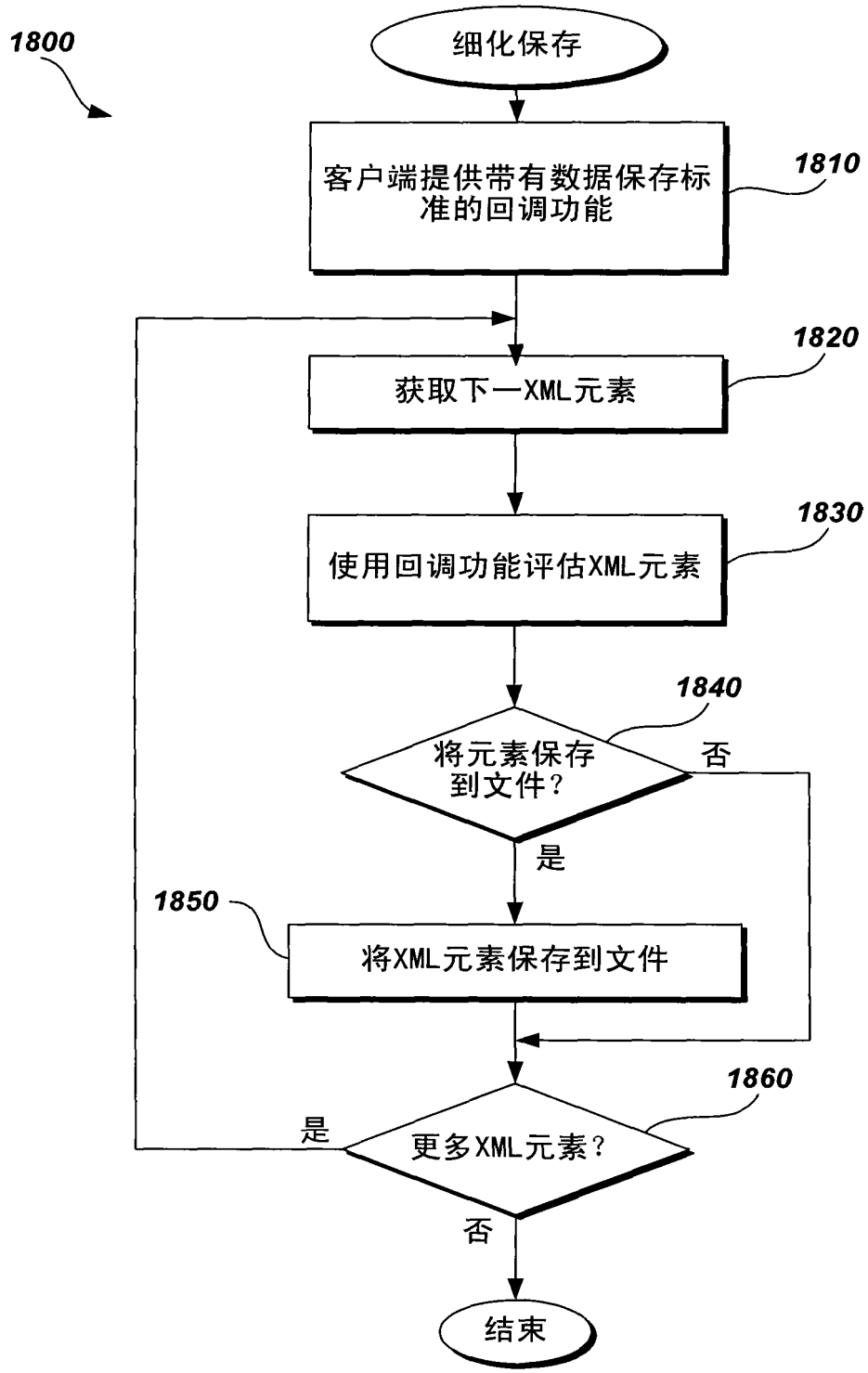

1800
细化保存
客户端提供带有数据保存标准的回调功能
1810
获取下一XML元素
1820
使用回调功能评估XML元素
1830
将元素保存到文件？
1840
否
是
1850
将XML元素保存到文件
更多XML元素？
1860
是
否
结束

图 18

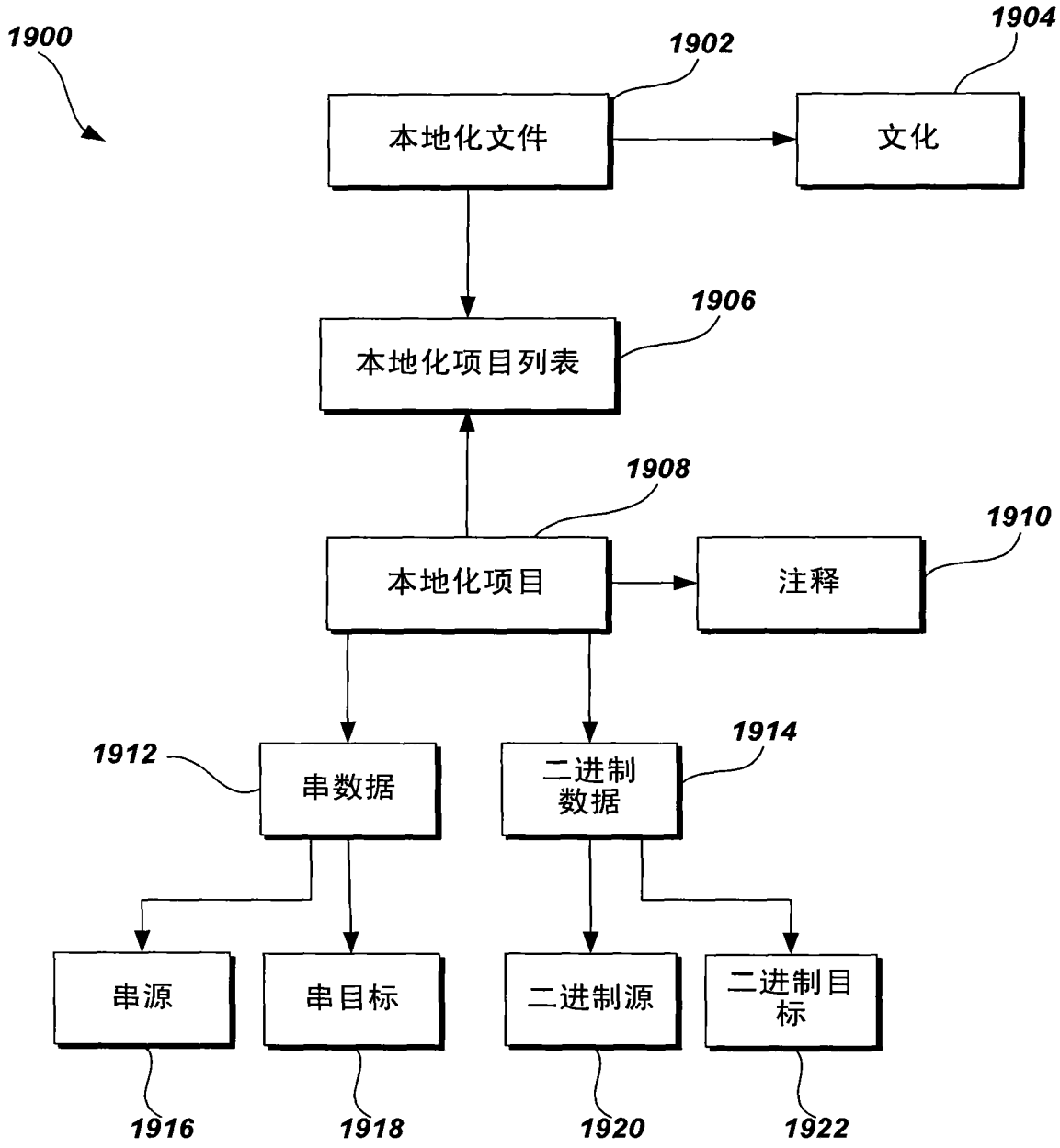

1900
1902
本地化文件
1904
文化
1906
本地化项目列表
1908
本地化项目
1910
注释
1912
串数据
1914
二进制数据
串源
串目标
二进制源
二进制目标
1916
1918
1920
1922

图 19

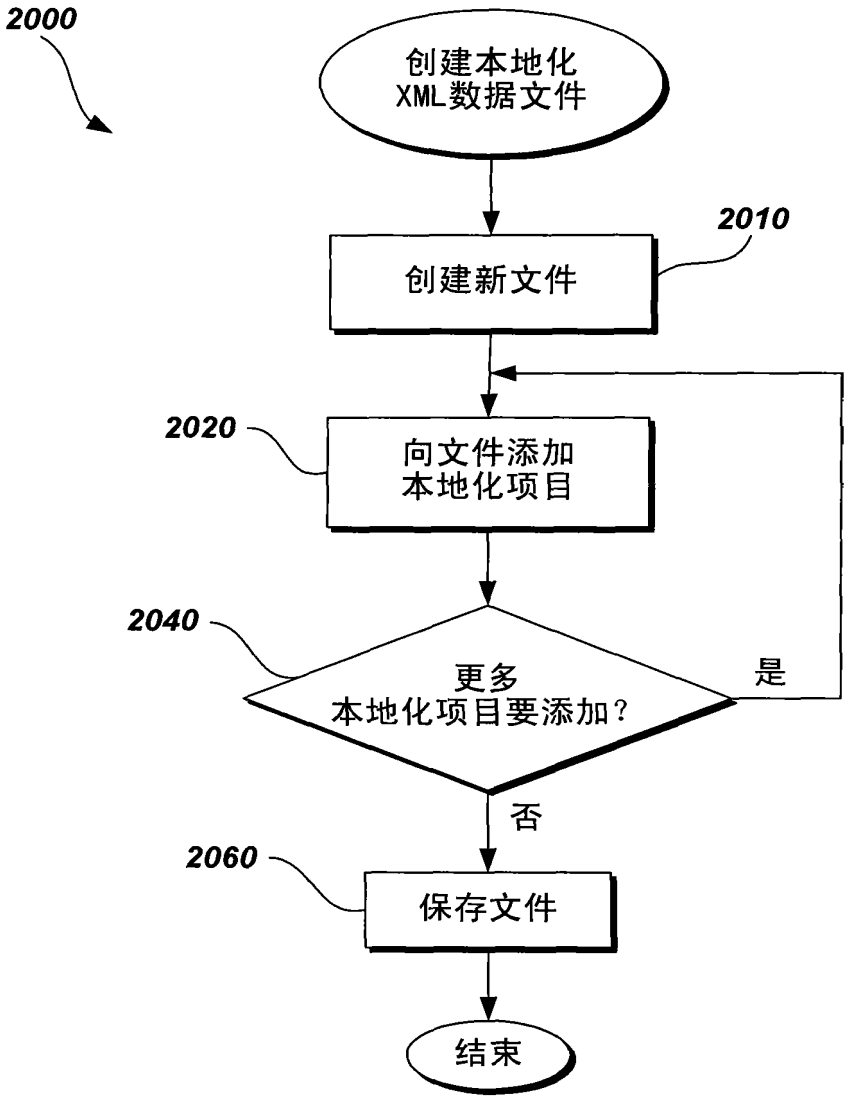

2000
创建本地化
XML数据文件
2010
创建新文件
2020
向文件添加
本地化项目
2040
更多
本地化项目要添加？
是
否
2060
保存文件
结束

图 20

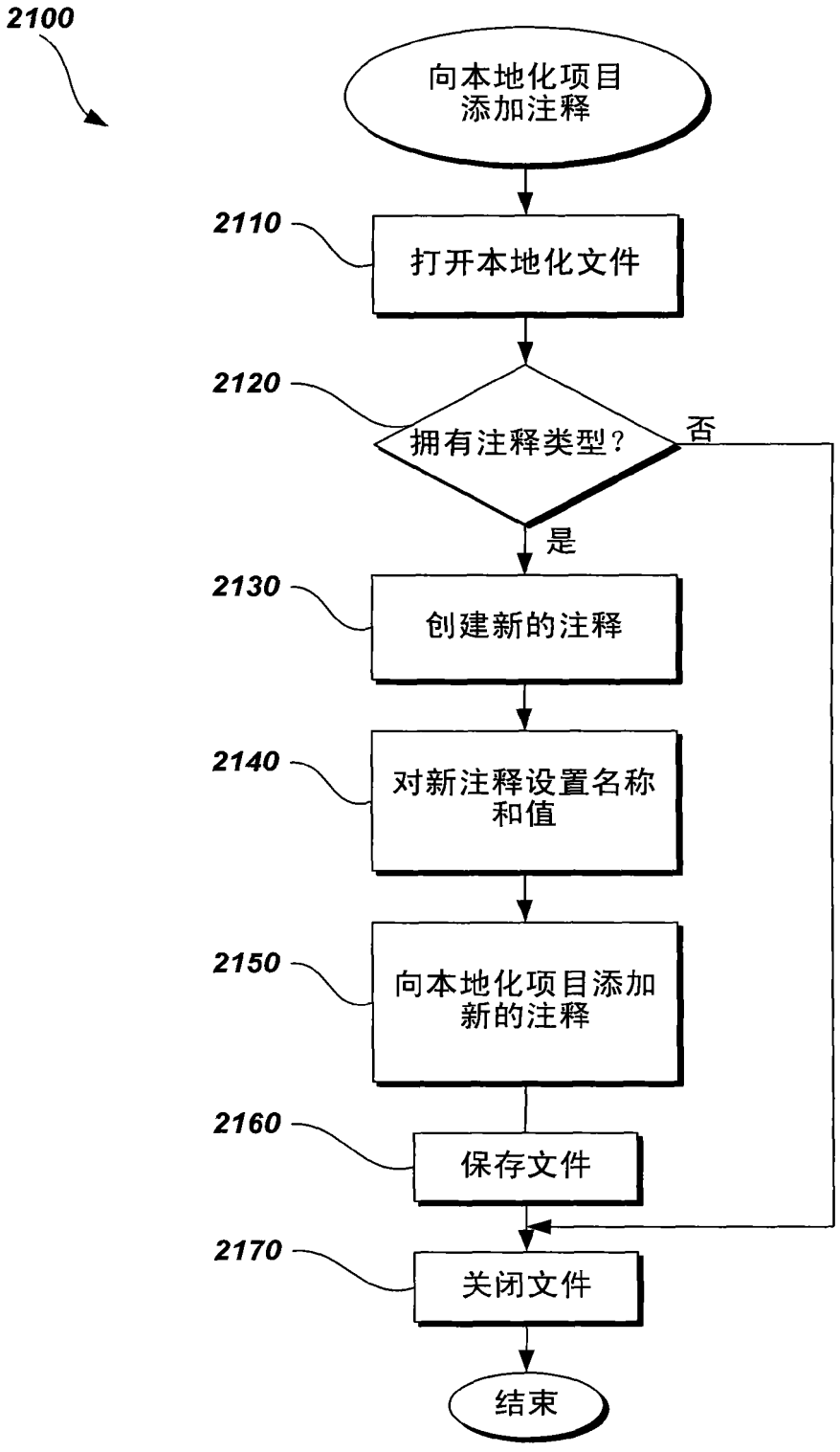

2100
向本地化项目添加注释
2110
打开本地化文件
2120
拥有注释类型？
否
是
2130
创建新的注释
2140
对新注释设置名称和值
2150
向本地化项目添加新的注释
2160
保存文件
2170
关闭文件
结束

图 21

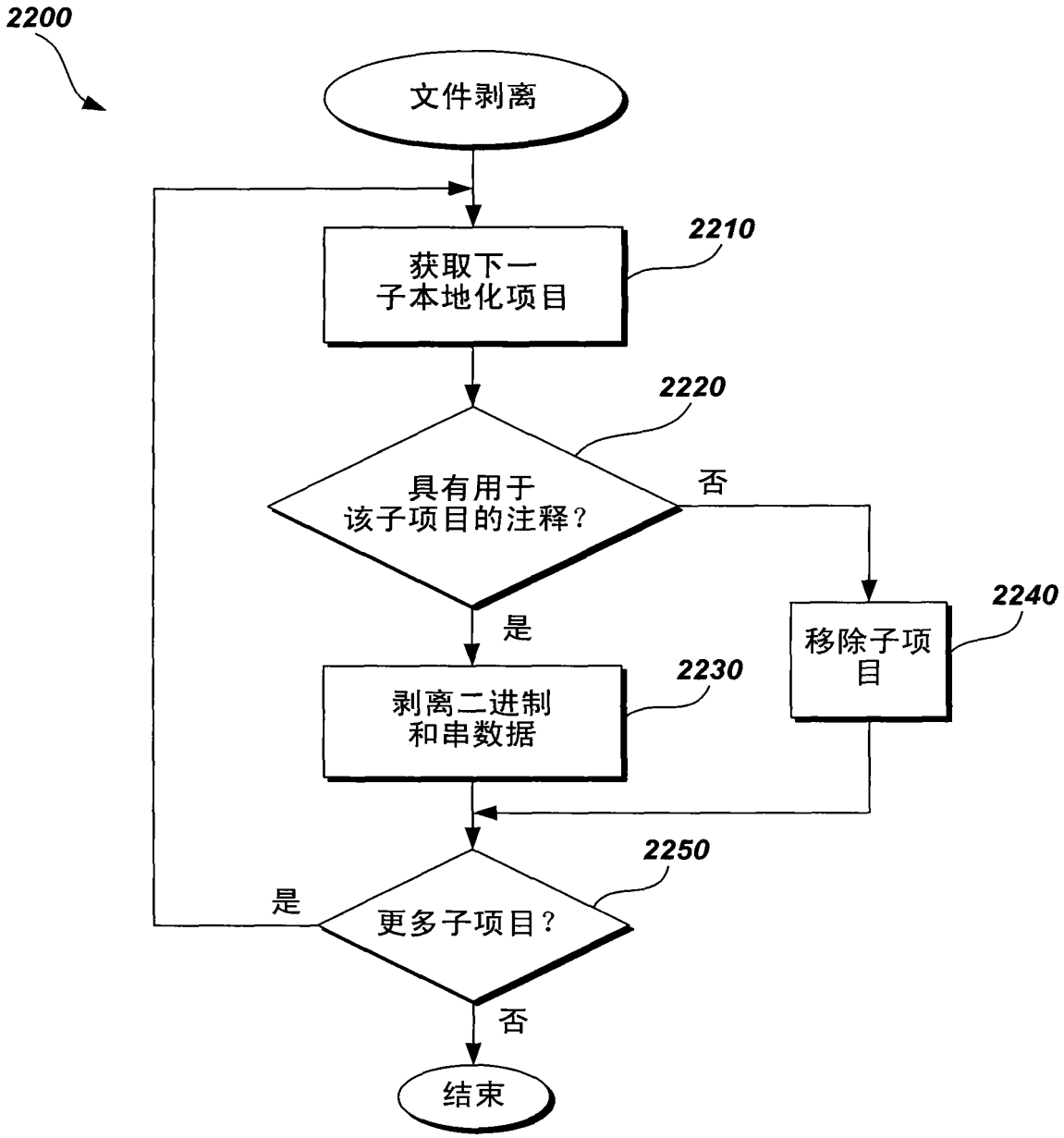

2200
文件剥离
获取下一
子本地化项目
2210
2220
具有用于
该子项目的注释？
否
是
2240
移除子项
目
2230
剥离二进制
和串数据
2250
是
更多子项目？
否
结束

图 22