



(12)发明专利

(10)授权公告号 CN 103049299 B

(45)授权公告日 2016. 11. 30

(21)申请号 201210331654.3

(22)申请日 2012.09.10

(65)同一申请的已公布的文献号
申请公布号 CN 103049299 A

(43)申请公布日 2013.04.17

(30)优先权数据
13/229697 2011.09.10 US

(73)专利权人 微软技术许可有限责任公司
地址 美国华盛顿州

(72)发明人 L.W.奥斯特曼 H.L.皮尔森
E.H.奥米亚 M.S.洛弗尔
M.普拉克里亚 S.C.罗维
T.H.巴苏 R.A.弗罗达茨克 曾炜
N.N.沃瓦 S.I.索尔卡
M.阿克西安金

(74)专利代理机构 中国专利代理(香港)有限公司 72001

代理人 刘红 汪扬

(51)Int.Cl.
G06F 9/45(2006.01)

(56)对比文件
CN 1920826 A,2007.02.28,
CN 101131672 A,2008.02.27,
US 2010/0169354 A1,2010.07.01,
US 6560613 B1,2003.05.06,

审查员 黄旭光

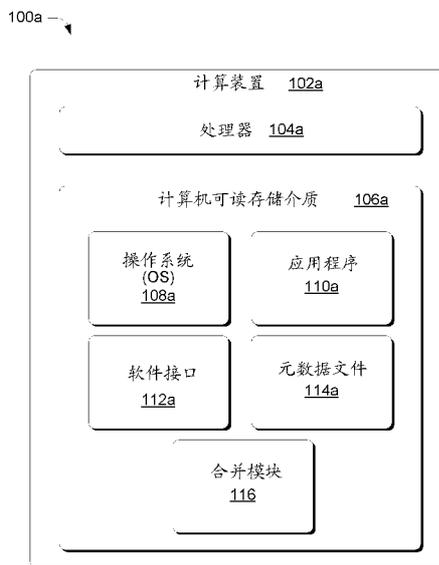
权利要求书2页 说明书10页 附图6页

(54)发明名称

灵活的元数据组合

(57)摘要

各种实施例提供了在多个类型系统之间对类型解析进行抽象的能力。至少一种类型可以在一个或多个可编程访问文件中描述。在一些实施例中,使用不同类型系统的应用程序可以在不了解类型的描述所在位置的情况下,可编程地访问并解析至少一个类型系统的类型。可替代地或附加地,至少部分地基于所述类型描述,可以对包含在一个或多个可编程访问文件中的类型描述进行分析,并且将其重新构成一个或多个新的可编程访问文件。



1. 一种计算机实施的用于类型解析的方法,包括:
 - 在多个文件中搜索(302)与正在被解析的类型相关的第一文件;
 - 响应于确定所述第一文件存在,发送(306)表明所述类型是命名空间的信息;
 - 响应于确定所述第一文件不存在,在所述多个文件中搜索(308)和与所述类型相关的第一命名空间级别层次相匹配的文件名;
 - 响应于确定和与所述类型相关的第一命名空间级别层次相匹配的文件名存在,处理(312)与所述文件名相关的文件以获得与所述类型相关的信息;
 - 响应于确定和与所述类型相关的所述第一命名空间级别层次相匹配的所述文件名不存在,确定(322)是否存在另一个与所述类型相关的命名空间层次级别;以及
 - 响应于确定另一个命名空间层次级别存在,在所述多个文件中搜索(318)与所述另一个命名空间层次级别相关的文件名。
2. 如权利要求1所述的方法,其中,在多个文件中搜索第一文件进一步包括:在所述多个文件搜索具有和与所述类型相关的名称相匹配的文件名的文件。
3. 如权利要求2所述的方法,其中,在多个文件中搜索第一文件进一步包括:在不了解所述类型所在位置的情况下,搜索所述第一文件。
4. 如权利要求1所述的方法,其中,该方法是在无用户介入的情况下执行。
5. 如权利要求1所述的方法,其中,所述多个文件包含元数据文件,其中单独的元数据文件包含与操作系统软件接口相关的描述。
6. 如权利要求5所述的方法,其中,所述元数据文件被配置成对独立于特定的编程语言的所述类型的描述。
7. 一种计算机实施的用于类型解析的方法,包括:
 - 在一个或多个元数据文件中搜索具有和与类型系统相关的类型的命名空间相匹配的文件名的文件;
 - 响应于确定所述文件存在,处理所述文件以获得与所述类型相关的信息;
 - 响应于确定所述文件不存在,执行至少一次:
 - 确定是否存在另一个与所述类型相关的命名空间层次级别;以及
 - 响应于确定另一个命名空间层次级别存在,在所述一个或多个元数据文件中搜索具有和所述另一个命名空间层次级别相匹配的文件名的文件;
 - 所述至少一次的执行响应于如下情况终止:
 - 确定具有和所述另一个命名空间层次级别相匹配的文件名的所述文件存在;或者
 - 确定另一个命名空间层次级别不存在。
8. 如权利要求7所述的方法,其中,在所述一个或多个元数据文件中搜索具有和所述另一个命名空间层次级别相匹配的文件名的文件包括:搜索和所述命名空间层次级别相匹配的部分文件名。
9. 如权利要求7所述的方法,其中,与类型系统相关的所述类型包括与操作系统相关的应用程序接口(API)。
10. 如权利要求7所述的方法,其中,在所述一个或多个元数据文件中搜索具有和与类型系统相关的类型的命名空间相匹配的文件名的文件进一步包括:在不了解所述类型所存在位置的情况下进行搜索。

11. 一种用于类型解析的计算机设备,包括:
用于在多个文件中搜索与正在被解析的类型相关的第一文件的组件;
用于响应于确定所述第一文件存在,发送表明所述类型是命名空间的信息的组件;
用于响应于确定所述第一文件不存在,在所述多个文件中搜索和与所述类型相关的第一命名空间级别层次相匹配的文件名的组件;
用于响应于确定和与所述类型相关的第一命名空间级别层次相匹配的文件名存在,处理与所述文件名相关的文件以获得与所述类型相关的信息的组件;
用于响应于确定和与所述类型相关的所述第一命名空间级别层次相匹配的所述文件名不存在,确定是否存在另一个与所述类型相关的命名空间层次级别的组件;以及
用于响应于确定另一个命名空间层次级别存在,在所述多个文件中搜索与所述另一个命名空间层次级别相关的文件名的组件。
12. 如权利要求11所述的设备,其中,在多个文件中搜索第一文件进一步包括:在所述多个文件搜索具有和与所述类型相关的名称相匹配的文件名的文件。
13. 如权利要求12所述的设备,其中,在多个文件中搜索第一文件进一步包括:在不了解所述类型所在位置的情况下,搜索所述第一文件。
14. 如权利要求11所述的设备,其中,该设备是在无用户介入的情况下操作的。
15. 如权利要求11所述的设备,其中,所述多个文件包含元数据文件,其中单独的元数据文件包含与操作系统软件接口相关的描述。
16. 如权利要求15所述的设备,其中,所述元数据文件被配置成对独立于特定的编程语言的所述类型的描述。
17. 一种用于类型解析的计算机设备,包括:
用于在一个或多个元数据文件中搜索具有和与类型系统相关的类型的命名空间相匹配的文件名的文件的组件;
用于响应于确定所述文件存在,处理所述文件以获得与所述类型相关的信息的组件;
用于响应于确定所述文件不存在,执行至少一次以下操作的组件:
确定是否存在另一个与所述类型相关的命名空间层次级别;以及
响应于确定另一个命名空间层次级别存在,在所述一个或多个元数据文件中搜索具有和所述另一个命名空间层次级别相匹配的文件名的文件;
所述至少一次的执行响应于如下情况终止:
确定具有和所述另一个命名空间层次级别相匹配的文件名的所述文件存在;或者
确定另一个命名空间层次级别不存在。
18. 如权利要求17所述的设备,其中,用于在所述一个或多个元数据文件中搜索具有和所述另一个命名空间层次级别相匹配的文件名的文件被进一步配置为搜索和所述命名空间层次级别相匹配的部分文件名。
19. 如权利要求17所述的设备,其中,与类型系统相关的所述类型包括与操作系统相关的应用程序接口(API)。
20. 如权利要求17所述的设备,其中,用于在所述一个或多个元数据文件中搜索具有和与类型系统相关的类型的命名空间相匹配的文件名的文件的组件被进一步配置为:在了解所述类型所存在位置的情况下进行搜索。

灵活的元数据组合

背景技术

[0001] 计算装置通常把运行操作系统作为管理计算装置的硬件和/或软件资源的一种方式。在一些情况下,操作系统可以提供对这些资源的简化的编程访问。例如,操作系统可以包含应用程序接口(API),用以暴露各种组件。只要应用程序获知与API相关的是什么类型,应用程序就可以利用与API不同的编程语言和/或类型系统成功地调用API。例如,API可以包含一个或多个输入和/或输出参数。为了调用API,程序员不仅决定API的参数,而且决定与这些参数相关的是什么数据类型。

[0002] 如上所述,可以用不同于调用编程语言类型系统的类型系统来描述API。为了连接不同的类型系统,程序员通常编写包装代码,用以在各类型系统之间进行转化。让程序员将API访问包含于程序中的一个方法是通过一个或多个文件和/或命名空间将API定义包含到源代码中。为了成功地将文件和/或命名空间结合到源代码,源代码可以被配置成包含对文件/命名空间的特定位置的引用(例如,硬编码的路径、利用该路径访问注册表项等)。如果该位置、文件名称、和/或命名空间名称发生变化,则链接被中断,直到用适当的修改对代码和/或软件工具进行更新。

发明内容

[0003] 提供这个发明内容部分来以简化的形式介绍下面在具体实施方式中进一步描述的概念的选择。这个发明内容并不旨在识别所请求保护的主题的关键特征或基本特征,也不旨在用于限制所请求保护的主题的范围。。

[0004] 各种实施例提供了在多个类型系统之间对类型解析进行抽象的能力。可以在一个或多个可编程访问文件中对至少一个类型进行描述。在一些实施例中,在不了解类型描述的所在位置的情况下,使用不同类型系统的应用程序可以可编程地访问并解析该类型系统的类型。可替代的或附加的,至少部分基于类型系统描述,可以对一个或多个可编程访问文件中所包含的类型描述进行分析并重新构成一个或多个新的可编程访问文件。

附图说明

[0005] 在所有附图中使用相同的数字来标记相同的特征。

[0006] 图1a示出了根据一个或多个实施例的、在其中可以应用本文中所描述的各种原理的操作环境。

[0007] 图1b示出了根据一个或多个实施例的、在其中可以应用本文中所描述的各种原理的操作环境。

[0008] 图2示出了根据一个或多个实施例的体系结构。

[0009] 图3示出了根据一个或多个实施例的流程图。

[0010] 图4示出了根据一个或多个实施例的关系图。

[0011] 图5示出了根据一个或多个实施例的流程图。

[0012] 图6示出了可以用以实施一个或多个实施例的示例性系统。

具体实施方式

[0013] 概述

[0014] 各种实施例提供了在多个类型系统之间对类型解析进行抽象的能力。使用一个类型系统的应用程序可以调用到第二个类型系统,只要该应用程序已了解如何在各类型系统之间连接。例如,类型系统的特征(例如数据类型、数据类型的行为、函数调用参数、事件等)可以在一个或多个可编程访问文件中描述。应用程序可以访问文件并且解析不同的类型系统。在一些实施例中,可以对类型解析进行抽象,使得在事先不了解访问哪个文件和/或文件所在位置的情况下,应用程序可以访问描述。

[0015] 在接下来的论述中,提供了标题为“操作环境”的章节并且描述了多个环境,在其中可以实施一个或多个实施例。在此之后,标题为“类型解析结构”的章节描述了可以进行可编程类型系统解析的体系结构。接下来,标题为“类型描述存储”的段落描述了可以用于实现类型描述的灵活存储的各种方法。最后,标题为“示例系统”的章节描述了一个可以用于实施一个或多个实施例的示例系统。

[0016] 前面已提供了对下文将要描述的各种实施例的概述,现在考虑示例操作环境,在其中可以实施一个或多个实施例。

[0017] 操作环境

[0018] 图1a和图1b示出了根据一个或多个实施例的操作环境,通常在100a和100b显示。图1a示出了参照一个或多个元数据文件的生成而可以利用的示例操作环境,如下所述。图1a的操作环境可以认为是“构建时”环境。图1b示出了参考灵活类型系统解析而可以利用的示例操作环境。图1b的操作环境可以看作是运行时环境。在一些实施例中,操作环境100a和100b具有至少一些类似的组件。因此,为简洁起见,图1a和图1b将被一起描述。与图1a相关的类似的组件将被标识为具有“1XXa”的命名约定的组件,而与图1b相关的组件将被标识为具有“1XXb”的命名约定的组件。类似地,特定于操作环境的组件将简单地标识为“1XX”。

[0019] 环境100a和100b分别包括:具有一个或多个处理器104a、104b,以及一个或多个计算机可读存储介质106a、106b的计算装置102a、102b。通过示例而非限制,计算机可读存储介质可以包括:与计算装置典型地相关的所有形式的易失性及非易失性存储器和/或存储介质。此类介质可以包括:ROM、RAM、闪速存储器、硬盘、可移动介质等。下面在图6中显示并描述了计算装置的一个具体示例。

[0020] 另外,计算装置102a、102b包含一个或多个操作系统(OS)108a、108b以及一个或多个应用程序110a、110b。操作系统108a、108b代表用于管理计算装置102a、102b的一个或多个软件和/或硬件资源的功能。这可以包括内存管理、文件管理、服务、功能、资源管理、外围设备管理等。一个或多个应用程序110a、110b代表被配置来在计算装置102a、102b中,通常在一个或多个操作系统108a、108b的协助下执行的软件。一个或多个应用程序110a、110b可以在与一个或多个操作系统108a、108b相同和/或不同的类型系统中执行,如下面进一步的描述。

[0021] 计算装置102a、102b也包括一个或多个软件接口112a、112b,这些软件接口代表对由软件和/或一个或多个应用程序所提供函数、服务、数据等的编程访问。一个或多个软件接口112a、112b可以实现对一个或多个操作系统108a、108b和/或一个或多个应用程序

110a、110b的编程访问。例如，一个或多个应用程序110a、110b可以通过调用一个或多个软件接口112a、112b来访问由一个或多个操作系统108a、108b所提供的功能。在一些实施例中，使用不同于软件接口的类型系统的类型系统的一个或多个应用程序112a、112可以可编程地解析类型差异，如下面进一步的描述。

[0022] 另外，计算装置102a、102b还包含一个或多个元数据文件114a、114b，这些元数据文件代表一个或多个机器可读文件，该机器可读文件包含与一个或多个软件接口112a、112b，一个或多个操作系统108a、108b，和/或一个或多个应用程序110a、110b相关的信息，诸如输入参数类型、参数调用顺序、接口之间的关系等。可替代的或附加的，一个或多个软件接口可以与连接到计算装置102a、102b的外围设备(诸如打印机、扫描仪、智能手机等)相关。在一些实施例中，一个或多个元数据文件114a、114b可以被配置成以任何合适的方式来描述一个或多个接口，如下面进一步的描述。

[0023] 计算装置102a还包括一个或多个合并模块116。一个或多个合并模块116代表可以阅读一个或多个元数据文件、对文件内容进行分析、以及输出一个或多个包含重新构成的内容的新元数据文件的内容的功能。在一些实施例中，可以至少部分地基于类型描述对内容进行重新组织。

[0024] 计算装置102b包括一个或多个类型解析模块118。一个或多个类型解析模块118代表配置来接收访问相关的类型数据的请求的功能，并且定位类型解析信息。在一些实施例中，一个或多个类型解析模块118可以在无用户输入的情况下，且独立于信息变化位置而定位类型解析信息。例如，当信息已从第一文件移动到第二文件时，一个或多个类型解析模块118可以在无用户输入的情况下定位接口描述信息，如下面进一步的描述。

[0025] 计算装置102a、102b可以体现为任何合适的计算装置，通过示例而非限制，诸如：台式计算机、便携式计算机、笔记本计算机、手持式计算机(例如个人数码助理(PDA))、移动电话等。

[0026] 上面已说明了示例操作环境，现在考虑对配置来实现独立于位置文件名称和/或位置的类型解析的类型解析结构加以论述。

[0027] 类型解析结构

[0028] 随着编程语言的技术进步，其能力也在进步。例如，用第一编程语言编写的应用程序可以调用到用第二编程语言编写的软件。本质上，编程语言具有不同的类型系统。因此，为了成功地调用位于不同的类型系统中的软件，第一编程语言利用多种技术来解析不同的类型。例如，程序员可以手动编写用于在各类型之间转换的包装代码。可替代地，类型系统定义可以保存在文件中，并且可以进行可编程访问。

[0029] 以编程方式访问一个包含类型定义的文件使应用程序能够确定运行时的能力和描述。然而，访问文件意味着不仅了解访问哪个文件，而且了解文件的位置。如果文件的内容改变和/或文件的位置改变，访问该文件的应用程序有可能会失败，除非适当地更新。这有时会在一个或多个应用程序与一个或多个文件之间形成非故意的耦合。

[0030] 各种实施例提供了在多个类型系统之间对类型解析进行抽象的能力。与类型解析相关的信息可以保存在可编程访问的文件中。在一些实施例中，采用一个类型系统的应用程序可以通过访问该文件而动态地解析第二类型系统。可替代地或附加的，应用程序可以在不了解访问哪个/哪些文件和/或该一个或多个文件所在位置的情况下访问该文件。

[0031] 考虑图2,该图示出了根据一个或多个实施例的示例体系结构200。体系结构200包含可以配置成在计算装置上执行的操作系统202。应当理解的是,为了简洁起见,图中未示出操作系统202的全部。操作系统202包括一个或多个操作系统组件204,操作系统组件204被配置来管理与计算装置相关的资源。在一些实施例中,一个或多个操作系统组件204可以提供对这些资源的编程访问、以及与管理这些资源相关的一个或多个服务和/或一个或多个特征。一个或多个操作系统组件204还可以包括与操作系统202相关的基本元素、以及从基本元素构建的复杂元素。虽然此示例示出了暴露由操作系统202所提供功能的体系结构,但应当领会和理解的是在不背离请所要求保护的的主题的精神的前提下,所述体系结构可以被应用于暴露由其它合适类型的应用程序所提供的功能。

[0032] 在一些实施例中,可以经由一个或多个接口(在这里图示为一个或多个操作系统接口206)暴露一个或多个操作系统组件204。这可以是任何适当形式的接口,诸如应用程序接口(API)。应用程序可以通过调用和/或执行一个或多个操作系统接口206以访问由一个或多个操作系统组件204所提供的功能,如下面进一步的描述。在一些情况下,应用程序利用不同于用于描述一个或多个操作系统接口206的类型系统的类型系统。在一些情况下,一个或多个操作系统接口206可以包括一个或多个应用程序二进制接口(ABI)。ABI以机器级描述用于调用功能、方法、API等的二进制合同(binary contract)。二进制合同可以包括与功能相关的标识或名称、可以用于调用函数的签名、传递给与参数相关的函数和/或数据类型的参数顺序,等。可替代地或附加地,二进制合同可以包括用于暴露与类型系统中的至少一种类型相关的行为的定义和/或规则。

[0033] 操作系统202还包含各种元数据208。元数据208可以包括类型解析信息,类型解析信息描述相关的一个或多个操作系统接口206的各种方面,诸如版本信息、什么方法是可用的、一个或多个接口采用什么参数、参数的数据类型、传递参数的顺序、数据类型行为和/或解析信息等。在一些实施例中,元数据可以包括与接口相关的层次信息,诸如描述一个或多个接口之间的关系的信息和/或以面向对象的方式描述该一个或多个接口的信息。元数据还可以包括类描述、类的相关的方法和参数等。在一些情况下,元数据可以对使用抽象类型系统(例如独立于特定编程语言的类型系统)的接口进行描述。可替代地或附加地,元数据可以包含对特定类型系统(诸如该抽象类型系统)的描述。一个或多个特定的编程语言转而是可以把特定的类型系统(例如该抽象类型系统)的描述映射到该一个或多个特定的编程语言中。此外,想要确定哪些接口是可用的程序员可以手动地和/或可编程地访问每个接口的描述。例如,为了确定一个或多个操作系统组件204存在哪些接口、接口在什么类型系统中被描述、以及如何调用它们,程序员可以访问相关的元数据208。

[0034] 体系结构200还包含一个或多个应用程序210。应用程序210可以包括由一个或多个编程语言(诸如HTML、JavaScript、Visual Basic、C#、C++等)所生成的一个或多个应用程序。在一些实施例中,应用程序210包含一个或多个对操作系统组件的调用。在一些情况下,应用程序210可以配置成首先可编程地确定哪个或哪些接口是可用的,然后调用一个或多个经确定的接口。在一些情况下,一个或多个应用程序210借助于一个或多个所生成的语言映射模块(language projection module)212而访问一个或多个接口,如下面进一步的描述。

[0035] 在一个或多个实施例中,生成的一个或多个语言映射模块212将抽象类型定义映

射至特定的编程语言。可以映射任何合适的编程语言,上文给出了其示例。在一些实施例中,生成的语言映射模块对于每种编程语言可以是唯一的。在其它实施例中,生成的语言映射模块可以是多用途的并且被多种编程语言所利用。映射实现了在无需额外的程序设计语句(例如包装功能)的情况下,特定编程语言能够对使用抽象类型系统所描述的当前和将来的接口访问。该映射进一步允许特定的编程语言能够以由特定编程语言产生的方式调用接口。任何合适类型的信息(诸如类、数据类型、函数指针、结构等)可以被映射。

[0036] 一个或多个接口的描述可以用于描述应用程序应该如何调用接口、以及描述与接口相关的类型系统的行为。响应于正在被定位的描述,应用程序可以动态地决定如何调用接口以及解析应用程序与接口之间的类型系统差异。在一些实施例中,使用不同类型系统的应用程序可以在不了解类型描述所在位置的情况下,可编程地访问并解析由接口所使用的类型。可替代地或附加地,如何对描述进行分组,可以使自动类型解析成为可能,如下面的进一步描述。

[0037] 考虑使用JavaScript编写应用程序的程序员。虽然此示例描述了使用JavaScript的实施例,但应当领会和理解的是在不背离所要求保护的的主题的精神的情况下可以使用任何编程语言。为了访问外部功能,程序员可以在源代码中包含被配置来识别与该功能相关的命名空间和/或类型的语句。例如,在源代码中可以包含和/或描述外部类型,如下:

[0038] `OperatingSystem.Foo.Bar.Type1`

[0039] 在这个具体示例中,Type1代表正在被访问的功能。这可以是任何合适类型的功能,在上文和下文中给出了其例子。上述语法代表可遍历地定位Type1的多级别命名空间层次。在最高级别中,Type1位于识别为“操作系统”的命名空间中。在“操作系统”内存在识别为“Foo”的命名空间。类似地,在“Foo”内存在识别为“Bar”的命名空间,Type1存在其中。因此,语法可以用于识别Type1的逻辑位置。然而,Type1的物理位置(例如,Type1信息存在于什么文件中或/或文件存在于什么目录中)有时可以变化。过去,可以通过对应用程序内的路径和文件名进行硬编码以确定物理位置(直接地或间接地)。因此,如果改变Type1信息的文件名和/或路径,那么对应用程序所使用的文件名称和/或路径的任何直接或间接引用将会是错误的,直到被更新。此外,应用程序有可能不能正常运行,直到直接或间接的引用被更新。

[0040] 各种实施例提供对类型解析进行抽象的能力。例如,可以在不确切了解相关类型解析信息位置的情况下,由应用程序对类型进行可编程解析。在一些实施例中,可以在不更新应用程序的情况下,利用具有新位置信息的解析信息对类型解析信息进行重新定位。应用程序可以配置为对类型解析信息进行定位而不管该类型解析信息存在于哪个文件中。

[0041] 考虑图3,其示出了根据一个或多个实施例的、描述方法中的步骤的流程图。所述方法可以由任何合适的硬件、软件、固件、或其组合执行。至少在一些实施例中,所述方法的各方面可以由在一个或多个计算装置中执行的一个或多个适当配置的软件模块来实施,如图1b的一个或多个类型解析模块118。在一些实施例中,所述方法可以在无用户介入的情况下执行。

[0042] 步骤302搜索具有与正在被解析的类型相匹配的文件名的文件。这可以包括搜索多个元数据文件,如上所述。参考上述语法说明,响应于应用程序访问`OperatingSystem.Foo.Bar.Type1`,步骤302搜索具有名称“`OperatingSystem.Foo.Bar.Type1`”的文件。任何合适类型的

搜索可以被执行。例如,搜索可以被配置来寻找类型与文件名之间的精确匹配、类型与文件名之间的部分匹配等。可替代地或附加地,搜索可以被配置来在一个目录、多个目录和/或子目录、或者其任意组合中进行搜索。

[0043] 步骤304确定该文件是否存在。响应于确定文件存在,步骤306发送表明该类型是命名空间的信息。响应于确定文件不存在,步骤308搜索和与正在被解析的类型相关的命名空间相匹配的文件名。在一些情况下,这可以包括操作字符串,该字符串包含用于确定在什么命名空间中进行搜索的命名空间层次。在上述示例中,Type1被图示为定位于命名空间层次中的向下的三个级别。字符串可以被操作以去除类型(例如Type1)并留下命名空间。这里,步骤308将会搜索与命名空间层次“OperatingSystem.Foo.Bar”相关的名称。与上述相类似,此搜索可以被配置来搜索精确匹配、部分匹配、在一个目录中的搜索、在子目录中搜索等。

[0044] 步骤310确定与该文件名相关的文件是否存在。响应于确定文件存在,步骤312通过对该文件进行处理以获得与该类型相关的信息。

[0045] 步骤314确定与类型相关的信息是否位于文件内。响应于确定与类型相关的信息是在文件内,步骤316返回该信息。例如,可以把该信息返回至调用程序。然而,响应于确定与该类型相关的信息不在文件内,则过程前进至步骤318。

[0046] 步骤318搜索和较高级别的命名空间相匹配的文件名。这可以是响应于确定文件不存在,例如响应于步骤310,或者响应于在文件内未定位到与类型相关的信息,例如响应于步骤314。可以以任何合适的方式来确定较高级别的命名空间。例如,在上述示例中,字符串可以被进一步操作来反映命名空间层次中的上一个级别(例如“OperatingSystem.Foo”)。步骤318确定较高级别的命名空间并且搜索具有相关名称的文件。

[0047] 步骤320确定与该文件名相关的文件是否存在。正如在步骤310的情况下,如果确定文件存在,过程前进至步骤312、314和/或316,其中对文件进行处理以获知相关的类型信息。

[0048] 响应于确定该文件不存在,步骤322确定是否存在另一命名空间层次级别。这可以以任意合适的方式来确定。例如,可以对字符串进行搜索以获得级别分隔符(level separator)。在上述实例中,“.”的语法在命名空间层次的级别之间区分。

[0049] 响应于确定另一命名空间层次级别存在,则过程重复,并且过程返回到步骤318以在新确定的命名空间内搜索文件。步骤318、320和322自身重复,直到找到适当文件或者已到达并搜索到命名空间层次的顶部。响应于确定另一命名空间层次级别不存在,步骤324返回错误。这可以包括抛出异常,显示弹出对话框等。

[0050] 通过上述方法的使用,可以对多个文件和/或位置进行搜索以获得类型解析信息。如图所示,类型搜索可以至少部分地基于层次命名空间信息。图3描述了类型搜索,该类型搜索开始于较低的命名空间层次级别(例如“OperatingSystem.Foo.Bar.Type1”)并且向上穿过每个命名空间级别,直到该类型被定位或者到达命名空间层次级别的顶部(例如“OperatingSystem”)。然而,可以按任意合适的顺序来执行此搜索。在一些实施例中,类型搜索可以开始于较高的命名空间层次级别(例如“OperatingSystem”)并且向下穿过每个命名空间级别,直到该类型被定位或者到达命名空间层次级别的底部(例如“OperatingSystem.Foo.Bar.Type1”)。在命名空间层次中搜索可以对类型解析信息可以存

在的位置进行抽象,并且还可以在不影响访问类型解析信息的应用程序的情况下,进一步使类型解析信息能够改变位置。

[0051] 上面已说明了被配置来实现独立于位置文件名称和/或位置的类型解析的类型解析体系结构,现在对根据一个或多个实施例的灵活的类型描述存储进行论述。

[0052] 类型描述存储

[0053] 元数据文件可以用于描述软件接口的各种方面。如上所述,元数据文件可以包括从类层次、抽象类型系统、相关方法、属性和事件等方面来描述接口的信息。在一些情况下,相关信息可以存在于多个文件中。例如,不同的元数据文件可以包括从属于相同命名空间的信息。虽然多个元数据文件可以给元数据文件的开发人员提供灵活性,但有时当进行搜索时却会阻碍元数据文件的用户。考虑这样一个例子,其中每个命名空间具有其自己的相关元数据文件。从划分的观点来看,各命名空间的单独文件可以将命名空间的改变与相关文件隔离开。然而,从搜索的观点来看,当在运行时进行搜索时,必须在多个文件中搜索信息会降低性能。此外,从获知的观点来看,随着文件数量增加,跟踪什么信息在什么文件中会被复杂化(compound)。

[0054] 各种实施例允许在不事先了解文件名称和类型解析信息所存在的文件位置的情况下进行类型解析。至少部分地基于组合规则,可以对一组输入文件的内容进行分析和划分。可以产生一组输出文件,并且将输出文件配置成包含经划分的内容。输出文件允许在不事先了解该内容所存在的位置的情况下定位该内容。

[0055] 作为示例,考虑图4,图中示出了一个或多个实施例的描述语言文件、元数据文件、及合并模块之间的关系。在至少一些实施例中,关系图中所示出的模块可以作为软件、硬件或者其任意组合而实施,例如图1a的一个或多个合并模块116。

[0056] 在图示和描述的实施例中,Foo.idl 402代表了被配置来描述一个或多个接口的描述语言文件。在一些实施例中,接口可以与操作系统和/或应用程序(诸如图1的操作系统接口108 和/或应用程序110)相关。描述语言文件可以利用任何合适的描述、置标语言、和/或语法(诸如,利用接口定义语言(IDL)、可扩展标记语言(XML)等)来描述接口。在此具体示例中,Foo.idl代表IDL文件。

[0057] Foo.idl 402包括三种类型的描述:Type Foo.Bar.Type1 404、Type Foo.Bar.Type2 406、和Type Foo.Bar.Type3 408。虽然用三个条目进行了说明,但应领会和理解的是在不背离所要求保护的的主题的精神的情况下任何合适数量的描述、以及各类型的描述都可以包含在Foo.idl 402中。例如,类型可以包括类、接口、方法、性质、事件等的描述。在此示例中,类型404、406和408被描述成包含在“Foo.Bar”的层次命名空间中。

[0058] 图4还示出了第二描述语言文件,Bar.idl 410。类似于Foo.idl 402, Bar.idl 410代表描述语言文件,该描述语言文件包括三种类型:Type Foo.Bar.Type4 412、Type Foo.Quux.Type1 414、和Type Foo.Quux.Type2 416。参照Type412、414和416的语法,Bar.idl 410在命名空间“Foo.Bar”中包含至少一个类型,在命名空间“Foo.Quux”中包含至少两个类型。

[0059] Foo.metadata 418代表至少部分地基于Foo.idl 402的机器可读的元数据文件。虽然未示出,但在一些实施例中,可以由编译器从Foo.idl 402中生成Foo.metadata 418。类似于Foo.idl 402, Foo.metadata 418包括对采用由元数据文件产生的格式的类型

Foo.Bar.Type1 420、Foo.Bar.Type2 422、和Foo.Bar.Type3 424的描述。Bar.metadata 426还代表机器可读元数据文件,该机器可读元数据文件至少部分地基于Bar.idl 410。如Foo.metadata 418的情况下,Bar.metadata 426包括类型描述Foo.Bar.Type4 428、Foo.Quux.Type1 430、和Foo.Quux.Type3 432。

[0060] 图4中包含的是合并模块434。合并模块434可以接收一个或多个输入文件,并且转而基于一组标准和/或规则生成一个或多个输出文件。例如,可以把输入命令和/或规则应用于合并模块434,合并模块434指定一个输入目录以在其中搜索输入的文件、一个输出目录以在其中放置输出文件、应用的验证级别、在什么级别的命名空间层次进行合并和/或划分、如何命名输出文件、生成输出文件的数量、应生成什么深度的元数据文件等。可以以任何合适的方式(诸如通过命令行和/或包含一个或多个规则的“生成文件”)向合并模块434提供标准和/或规则。

[0061] 在此示例中,Foo.metadata 418和Bar.metadata 426是合并模块434的输入。合并模块434解析输入文件,并且按照指令生成一个或多个输出文件。这里,合并模块434生成两个输出文件:Foo.Bar.metadata 436和Foo.Quux.metadata 438。Foo.Bar.metadata 436包括从属于“Foo.Bar”命名空间(例如Foo.Bar.Type1 440、Foo.Bar.Type2 442、Foo.Bar.Type3 444和Foo.Bar.Type446)的类型。类似地,Foo.Quux.metadata 438包括从属于“Foo.Quux”命名空间(例如,Foo.Quux.Type1 448、Foo.Quux.Type2 450)的类型。正如在上述情况下的,Foo.Bar.metadata 436和Foo.Quux.metadata 438代表机器可读元数据文件,该机器可读元数据文件可能包含来自相关输入文件的经重新组织和/或经重新分组的数据。因此,合并模块434可以根据一组标准来分析来自多个文件的内容并重新构成该内容。

[0062] 在一些实施例中,文件命名约定可以被利用来实现抽象类型解析。例如,命名空间层次可以用作命名约定。在图4中,文件Foo.Bar.metadata 436和Foo.Quux.metadata 438在它们的文件名称(例如分别“Foo.Bar”和“Foo.Quux”)中包括相关的命名空间层次。通过采用此命名约定,可以基于其相关的命名空间而不是特定的文件名来搜索类型信息。可替代地或附加地,输出文件可以包含多个级别的命名空间层次数据。在图4中, Foo.Bar.metadata 436被图示为“Foo.Bar”命名空间中包含的数据。然而, Foo.Bar.metadata 436也可以配置成包含存在于“Foo.Bar.Level1. Level2. Level3”的命名空间层次级别中的类型信息。因此,命名约定表明可以包含在文件内的最高级别的命名空间。

[0063] 另外,可以利用搜索算法(例如图3中所描述)来穿过不同级别的层次命名空间(及相关的文件),直到适当的类型被定位。通过使在相同命名空间层次级别中的所有类型信息存在于相同文件中,可以在不影响利用该信息的应用程序的情况下使与特定级别相关的信息在各文件之间移动。可以以任何合适的方式来划分与命名空间层次相关的信息。例如,多个层次级别可以存在于相同文件(例如 Foo.Bar、Foo.Bar.Level1 和Foo.Bar.Level2,均存在于名为“Foo.Bar.metadada”的文件中)中,或者每个文件可以具有其自己的相关的文件(例如,处于Foo.Bar级别的信息存在于“Foo.Bar.metadata”中,处于Foo.Bar.Level1的信息存在于“Foo.Bar.Level1.metadada”文件中等)。如果信息是根据其相关的命名空间层次来放置的,则可以通过去除对特定文件名称的相关性来对类型解析进行抽象。

[0064] 现在考虑图5,图中示出了描述根据一个或多个实施例的方法中的步骤的流程图。该方法可以由任何合适的硬件、软件、固件、或者其组合实施。至少在一些实施例中,该方法的各方面可以由在一个或多个计算装置(如图1a的合并模块116)中执行的一个或多个适当配置的软件模块实施。

[0065] 步骤502接收与生成一个或多个输出文件相关的一个或多个输入标准。在一些实施例中,所述输入标准可以包括规定如何把信息划分成一个或多个输出文件的一个或多个规则。例如,所述输入标准可以指定一个或多个命名空间层次级别以一起组合到输出文件中。可替代地或附加地,所述输入标准可以描述在哪里找到信息,比如通过规定从哪个文件和/或目录中获得信息。

[0066] 响应于接收所述一个或多个输入标准,步骤504接收包含与一个或多个软件接口相关的信息的一个或多个输入文件。所述信息可以是任何合适类型的信息,比如:对使用抽象类型系统的软件接口的描述、面向对象的联系、方法、性质、函数指针、输入和/或输出参数、类型系统信息等。在一些实施例中,类型信息可以包括命名空间层次信息。

[0067] 响应于接收一个或多个输入文件,步骤506至少部分地基于该一个或多个输入标准对信息进行分析。这可以包括对一个或多个输入文件进行分析从而确定各文件内存在什么命名空间层次类型信息。

[0068] 响应于对信息的分析,步骤508生成至少一个包含所述信息的输出文件。在一些实施例中,所述输出文件可以包含已被划分成不同文件的信息,这些文件至少部分地基于该输入标准。输出文件可以以任何合适的方式配置,其示例提供如上。例如,一个或多个输出文件可以被配置成以命名约定来命名的元数据文件,该命名约定基于包含在该元数据文件中的命名空间层次。可替代地或附加地,一个或多个输出文件可以被配置成包含多个级别的命名空间层次。通过把命名空间层次信息一同分组到输出文件,应用程序可以基于命名空间层次级别而不是特定的文件名来搜索信息。这使得在不影响应用程序的情况下对信息进行灵活划分成为可能。只要应用程序了解特定的类型存在于哪个命名空间层次级别中,相关的类型信息就可以被置于输出文件中,而不管如何对命名空间层次级别进行分组。

[0069] 上面已说明了灵活的类型描述存储,现在对根据一个或多个实施例的示例系统进行论述。

[0070] 示例系统

[0071] 图6示出了可以用于实施上述各种实施例的示例计算装置600。例如,计算装置600可以是图1a和图1b的计算装置102a和/或102b或者任何其它合适的计算装置。

[0072] 计算装置600包括一个或多个处理器或处理单元602、一个或多个存储器和/或存储组件604、一个或多个输入/输出(I/O)装置606、和使各种组件和装置相互通信的总线608。总线608代表若干类型的总线结构中任一类型的一个或多个,其包括内存总线或内存控制器、外围总线、图形加速端口、以及处理器、或者采用多种总线结构中的任何总线结构的局部总线。总线608可以包括有线总线和/或无线总线。

[0073] 存储器/存储组件604代表一个或多个计算机硬件存储介质。组件604可以包括易失性介质(诸如,随机存取存储器(RAM))和/或非易失性介质(诸如只读存储器(ROM)、闪存存储器、光盘、磁盘等等)。组件604可以包括固定介质(诸如,RAM、ROM、固定硬盘驱动器等)以及可移动介质(例如,闪存存储器驱动器、移动硬盘驱动器、光盘等等)。

[0074] 一个或多个输入/输出装置606允许用户把命令和信息输入至计算装置600,并且还允许将信息提供给该用户和/或其它组件或装置。输入装置的例子包括:键盘、光标控制装置(例如,鼠标)、传声器、扫描仪等等。输出装置的示例包括:显示装置(例如,监视器或投影仪)、扬声器、打印机、网卡,等等。

[0075] 可以在软件或程序模块的一般上下文中描述本文各种技术。一般来说,软件包括执行特定任务或者实施特定抽象数据类型的例行程序、程序、对象、组件、数据结构等等。这些模块和技术的实施方式可以存储在某种形式的计算机可读介质中或者通过其传输。计算机可读介质可以是可以被计算装置访问的任何可用的媒介或介质。通过示例而并非限制,计算机可读介质可以包括“计算机可读存储介质”。

[0076] “计算机可读存储介质”包括以用于信息存储的任何方法或技术实施的易失性和非易失性、可移动和不可移动硬件介质,诸如计算机可读指令、数据结构、程序模块、或其它数据。计算机可读硬件存储介质包括但不限于:RAM、ROM、EEPROM、闪速存储器或者其它存储技术、CD-ROM、数字多用途光盘(DVD)或其它光存储器、磁带盒、磁带、磁盘存储器或其它磁存储装置、或者可以用于存储期望的信息并且可以被计算机访问的任何其它介质。

[0077] 结论

[0078] 各种实施例提供在多个类型系统之间对类型解析进行抽象的能力。至少一个类型可以在一个或多个可编程访问文件中描述。在一些实施例中,采用不同类型系统的应用程序可以在不了解类型的描述所在位置的情况下可编程地访问并且解析至少一个类型系统的类型。可替代地或附加地,至少部分地基于所述类型描述,可以对包含于一个或多个可编程访问文件中的类型描述进行分析并且重新构成一个或多个新的可编程访问的文件。

[0079] 尽管已以特定于结构特征和/或方法学操作的语言描述该主题,但要理解的是所附权利要求中所定义的主题不必局限于上述具体特征或操作。相反,上述具体特征和操作是以实施权利要求的实例形式而公开的。

100a

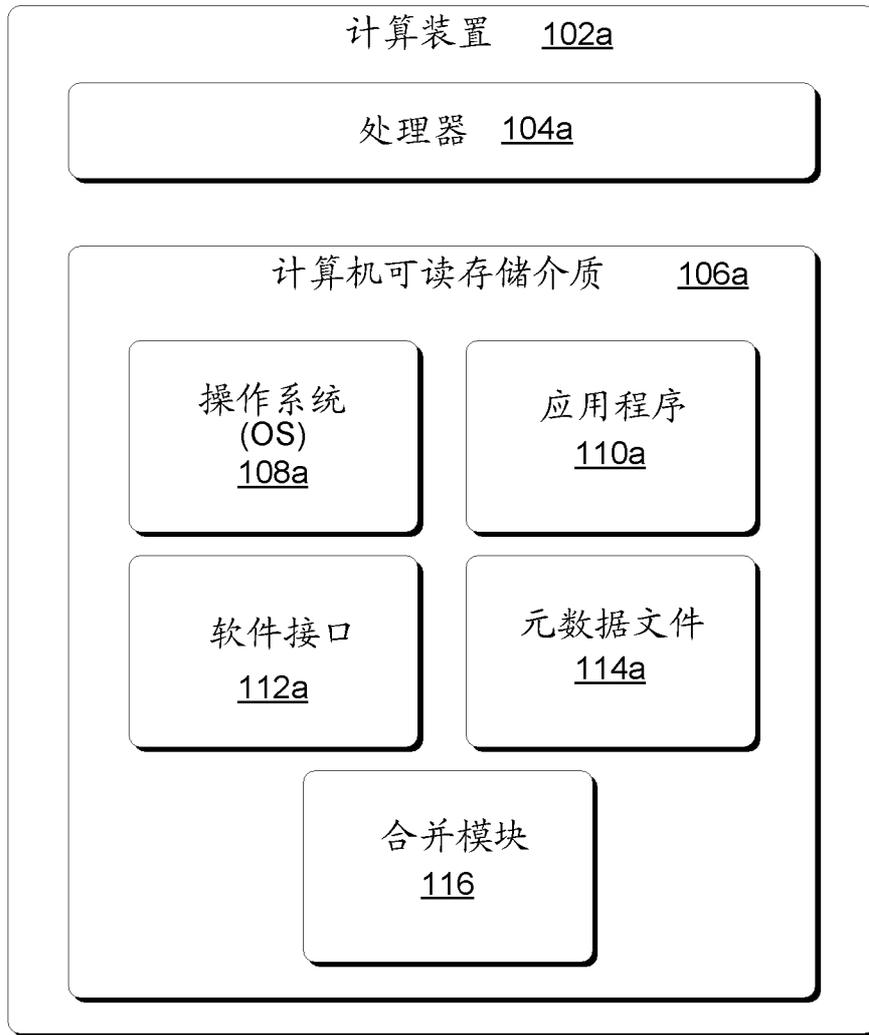


图 1a

100b

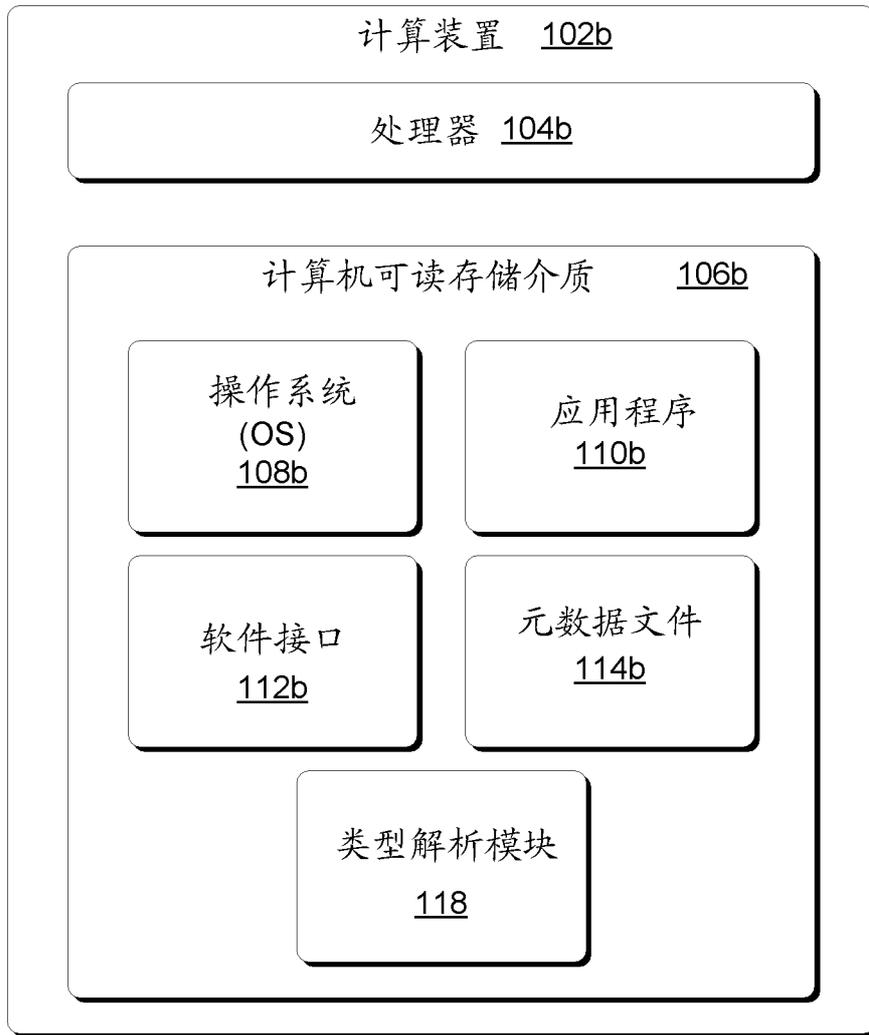


图 1b

200

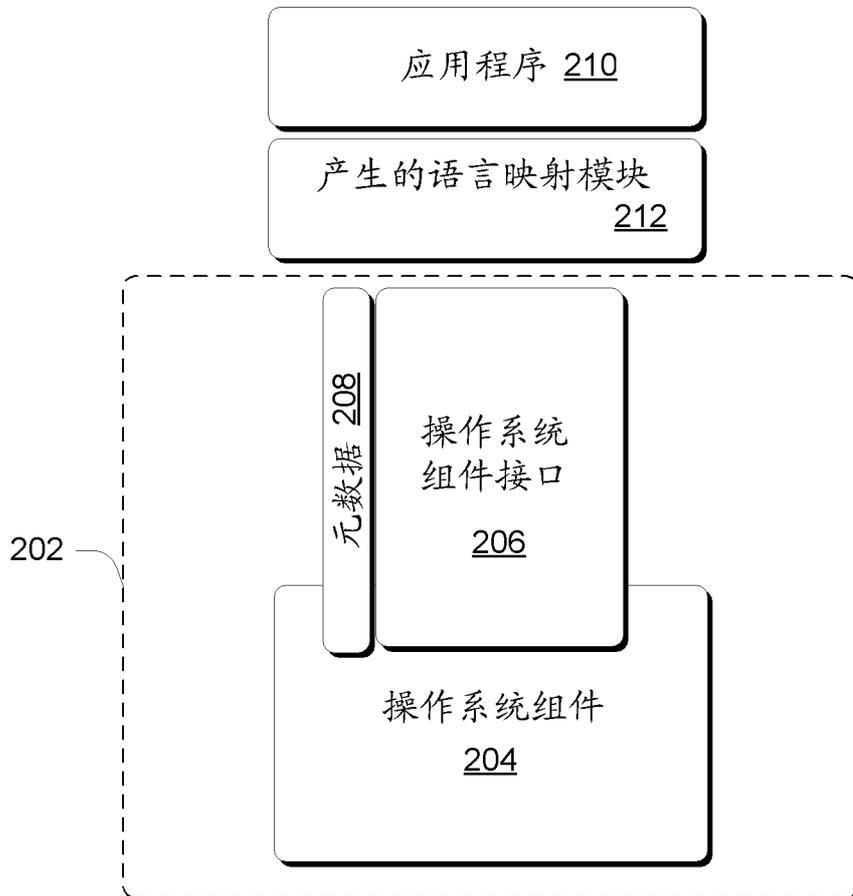


图 2

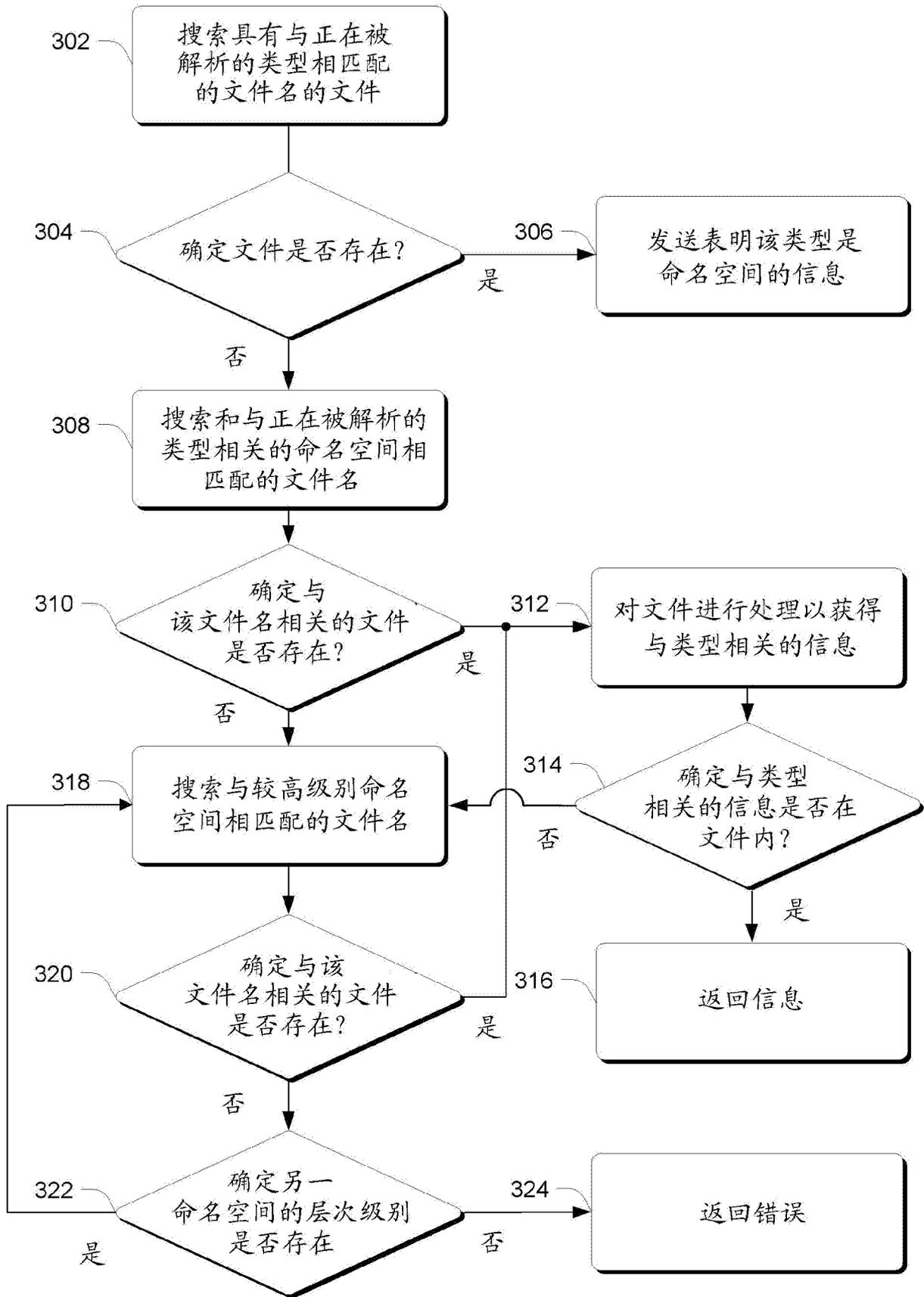


图 3

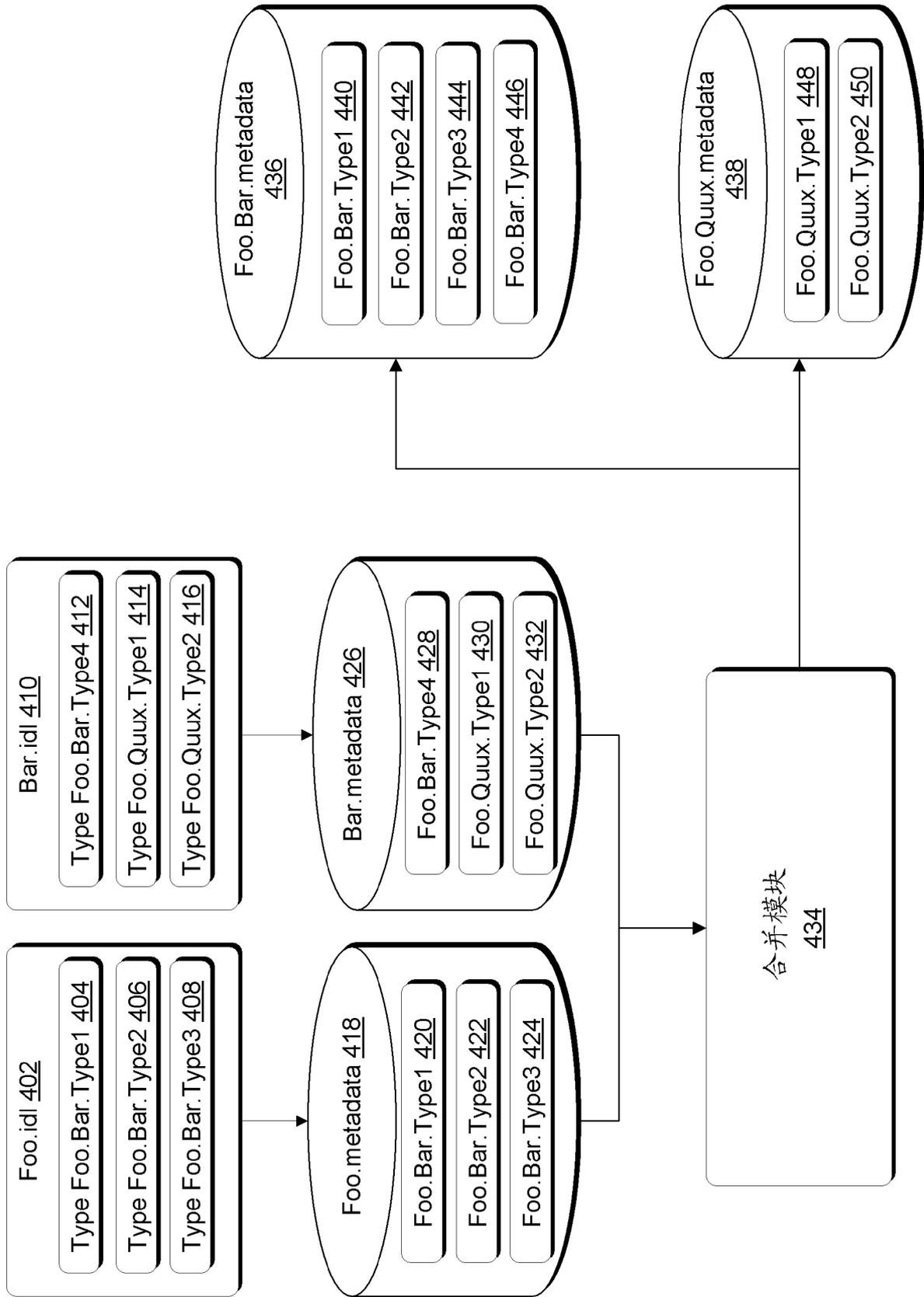


图 4

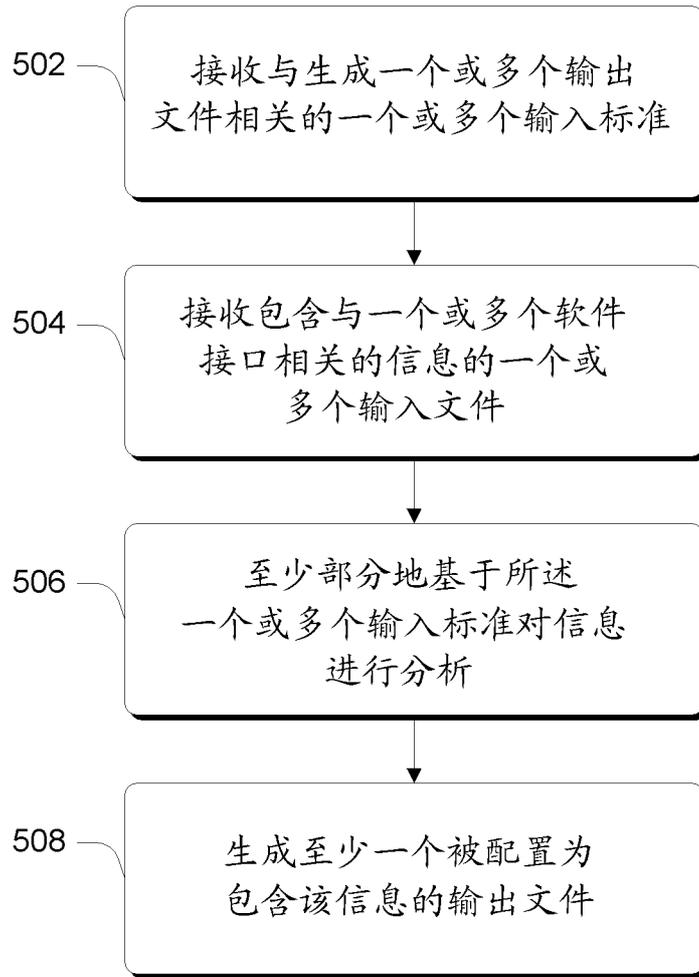


图 5

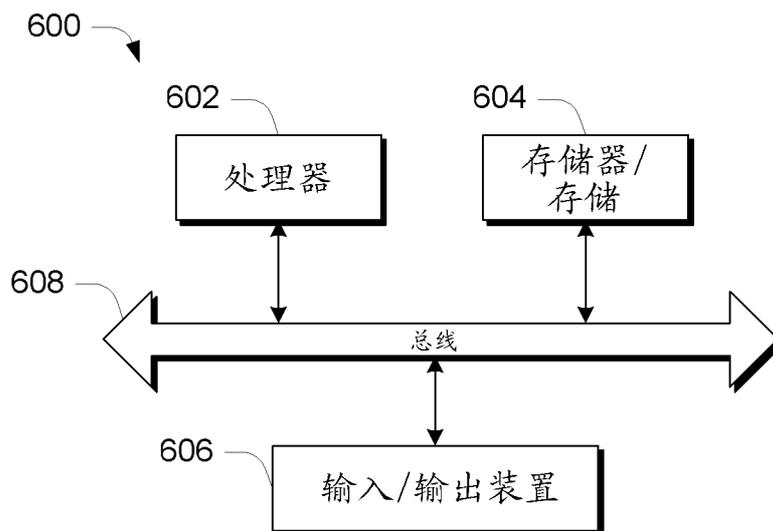


图 6