(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 107810474 B
(45)授权公告日 2020.12.29

(21)申请号 201680037594.0
(22)申请日 2016.06.27
(65)同一申请的已公布的文献号
申请公布号 CN 107810474 A
(43)申请公布日 2018.03.16
(30)优先权数据
14/796,695 2015.07.10 US
(85)PCT国际申请进入国家阶段日
2017.12.26
(86)PCT国际申请的申请数据
PCT/US2016/039561 2016.06.27
(87)PCT国际申请的公布数据
WO2017/011176 EN 2017.01.19
(73)专利权人 谷歌有限责任公司
地址 美国加利福尼亚州
(72)发明人 卡尔米·格鲁什科
(74)专利代理机构 中原信达知识产权代理有限责任公司 11219
代理人 李宝泉 周亚荣
(51)Int.Cl.
G06F 8/41(2018.01)
G06F 8/74(2018.01)
(56)对比文件
CN 101836188 A,2010.09.15
CN 101968736 A,2011.02.09
US 2011035729 A1,2011.02.10
US 2014201709 A1,2014.07.17
审查员 庄文龙

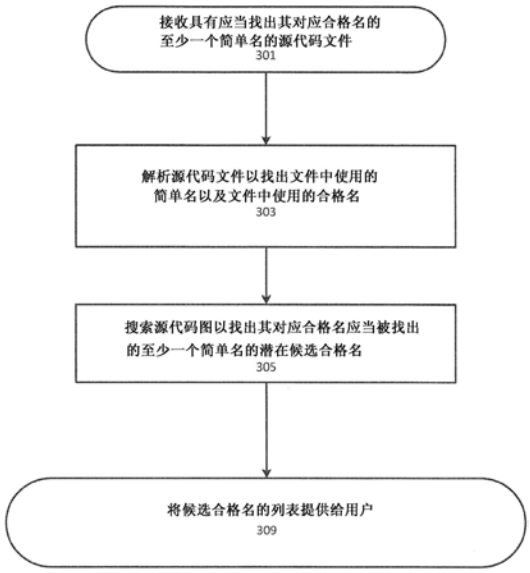
权利要求书2页 说明书5页 附图3页

(54)发明名称

大规模源代码存储库中的自动导入及依赖性

(57)摘要

本发明公开了用于找出源代码文件中的至少一个简单名的合格名的候选的系统和方法。可以接收具有需要对应合格名的至少一个简单名的源代码文件。可以解析所述源代码文件以确定所述文件中的简单名和所述文件中的合格名。针对需要对应合格名的至少一个简单名,可以搜索源代码图以找出与所述至少一个简单名相对应的潜在候选合格名。在从所述源代码图中接收到潜在候选合格名的列表后,可以向用户输出所述列表以便所述用户选择适当的合格名。针对所述合格名,还可以选择构建依赖性。



CN 107810474 B

1.一种用于找出待编译的源代码文件中的至少一个简单名的合格名的候选的计算机实现的方法,所述方法包括:

接收待编译的源代码文件,所述源代码文件包括对应合格名被编译器需要的至少一个简单名;

解析所述源代码文件以确定所述源代码文件中的简单名和合格名;

针对对应合格名被需要的至少一个简单名,搜索源代码图以找出与所述至少一个简单名相对应的潜在候选合格名;

在接收到来自所述源代码图的潜在候选合格名的列表后,用附加信息充实潜在候选合格名的所述列表,其中,充实潜在候选合格名的所述列表包括:

确定为了构建与潜在候选合格名的所述列表中所包括的潜在候选合格名相对应的合格名包向构建过程添加的源代码所增加的时间或资源;

将所述合格名包的名称添加到潜在候选合格名的所述列表中;

输出所述列表;

一旦从潜在候选合格名的所述列表中选择了合格名,则确定特定合格名的构建依赖性以及构建目标;以及

基于所确定的构建依赖性以及构建目标来编译源代码。

2.根据权利要求1所述的计算机实现的方法,其中,充实所述列表包括:

在输出所述列表之前,充实所述列表以包括其他源代码文件已经使用特定类型的潜在候选合格名的次数。

3.根据权利要求1所述的计算机实现的方法,其中,充实所述列表包括:

基于源代码存储库的其他部分中已经使用特定潜在候选合格名的频率来排名所述列表。

4.根据权利要求1所述的计算机实现的方法,进一步包括:

如果所述列表仅包含一个潜在合格名,则无需用户交互便自动选择所述合格名。

5.根据权利要求1所述的计算机实现的方法,进一步包括:

提供接口以接收对潜在候选合格名的所述列表中的合格名的选择。

6.根据权利要求1所述的计算机实现的方法,其中,充实所述列表包括:

确定合格名包是否从其中与所述合格名包潜在相对应的简单名当前正被引用的源代码文件可见;以及

响应于确定所述合格名包从所述源代码文件可见,提供所述合格名包的名称作为潜在候选合格名。

7.一种用于找出待编译的源代码文件中的至少一个简单名的合格名的候选的系统,所述系统包括:

一个或多个处理设备;以及

一个或多个存储指令的存储设备,所述指令在由所述一个或多个处理设备执行时使得所述一个或多个处理设备:

接收待编译的源代码文件,所述源代码文件包括对应合格名被编译器需要的至少一个简单名;

解析所述源代码文件以确定所述源代码文件中的简单名和合格名;

针对对应合格名被需要的至少一个简单名,搜索源代码图以找出与所述至少一个简单名相对应的潜在合格名;

在接收到来自所述源代码图的潜在候选合格名的列表后,用附加信息充实潜在候选合格名的所述列表,其中,充实潜在候选合格名的所述列表包括:

确定为了构建与潜在候选合格名的所述列表中所包括的潜在候选合格名相对应的合格名包向构建过程添加的源代码所增加的时间或资源;

将所述合格名包的名称添加到潜在候选合格名的所述列表中;

输出所述列表;

一旦从潜在候选合格名的所述列表中选择了合格名,则确定特定合格名的构建依赖性以及构建目标;以及

基于所确定的构建依赖性以及构建目标来编译源代码。

8.根据权利要求7所述的系统,其中,充实所述列表包括:

在输出所述列表之前,充实所述列表以包括其他源代码文件已经使用特定类型的潜在候选合格名的次数。

9.根据权利要求7所述的系统,其中,充实所述列表包括:

基于源代码存储库的其他部分中已经使用特定潜在候选合格名的频率来排名所述列表。

10.根据权利要求7所述的系统,进一步包括:

如果所述列表仅包含一个潜在合格名,则无需用户交互便自动选择所述合格名。

11.根据权利要求7所述的系统,进一步包括:

提供接口以接收对潜在候选合格名的所述列表中的合格名的选择。

12.根据权利要求7所述的系统,其中,充实所述列表包括:

确定合格名包是否从其中与所述合格名包潜在相对应的简单名当前正被引用的源代码文件可见;以及

响应于确定所述合格名包从所述源代码文件可见,提供所述合格名包的名称作为潜在候选合格名。

大规模源代码存储库中的自动导入及依赖性

背景技术

[0001] 开发者经常构建包含外部代码或库的复杂软件应用。例如，开发者可能希望在其应用中包括人员列表。开发者可能代替定义他自己的列表对象而包括来自外部包的列表类。在如图1所示的这个示例中，开发者可以在他的代码中包括来自标准库的列表，诸如“java.util.List”。“java.util.List”是通过java.util包可公开获得的有序集合类。一旦列表被导入开发者的源代码文件中，开发者便可以简单地利用简单名“List”来引用该列表，而不必在各处实例化或使用该列表时使用完全合格名(qualified name)“java.util.List”。例如，在图1的行3中，代码仅涉及简单名“List”，而不涉及合格名“java.util.List”。

[0002] 虽然使用标准集成开发环境(IDE)能够使定义合格名的包、库或外部代码导入源代码的过程对于小规模代码库简单直接，但大规模代码库对于标准IDE而言具有过多外部依赖性而无法有效率地找出称为合格名的适当全名或者依赖性的适当构建系统构造。

[0003] 常规的系统通过查看整个代码库并且试图确定适当的包、库或其他外部代码来尝试实时找出合格名。这个过程非常耗时。使用这种方法根本无法在合理的时间内处理大量的源代码。

[0004] 正如本发明人所认识到，期望能够快速地找出合格名的候选以导入适当的包、库或外部代码，以用于源代码中存在的简单名引用以及提供合格名的构建系统依赖性。

发明内容

[0005] 本说明书描述有关针对源代码文件——例如Java语言的源代码文件——中的至少一个简单名找到合格名的候选的技术。

[0006] 一般而言，本说明书中所述的主题的一方面能够被实施在一种用于找出源代码文件中的至少一个简单名的合格名的候选的系统中。一种示例系统可以包括一个或多个处理设备以及一个或多个存储设备，所述存储设备存储实现示例方法的指令。一种示例方法可以包括：接收具有需要对应合格名的至少一个简单名的源代码文件；解析所述源代码文件以确定所述文件中的简单名和所述文件中的合格名；针对需要对应合格名的至少一个简单名，搜索源代码图以找出与所述至少一个简单名相对应的潜在候选合格名；以及在接收到来自所述源代码图的潜在候选合格名的列表后，输出所述列表。

[0007] 这些及其他实施例能够可选地包括以下特征中的一个或多个：在输出所述列表之前，可以用附加信息过滤或充实所述列表；一旦合格名被选择，便可以确定特定合格名的构建依赖性以及构建目标；在输出所述列表之前，可以基于其他源代码文件已经使用特定类型的合格名的次数来过滤所述列表；可以基于源代码存储库的其他部分中已经使用特定合格名的频率来排名所述列表；如果所述列表仅包含一个潜在合格名，则无需用户交互便可以自动选择该名作为所述合格名；可以为用户提供接口以接收对潜在候选合格名的列表中的合格名的选择；过滤可以包括：确定合格名包是否从其中与所述合格名潜在相对应的简单名当前正被引用的源代码文件可见，以及响应于确定所述合格名包从所述源代码文件可

见，提供所述合格名包的名称作为潜在候选合格名；以及过滤可以包括：确定合格名包能够向构建过程增添的复杂度，包括为了构建所述源代码文件所增加的时间和/或资源，以及响应于确定合格名包向所述构建过程增添的复杂度是可承受的量，将所述合格名包的名称添加到潜在候选合格名的列表。

附图说明

[0008] 图1是包括合格名和简单名的源代码的示例。

[0009] 图2是图示出用于确定源代码文件中的简单名的合格名的示例系统的框图。

[0010] 图3是图示出示例计算设备的框图。

具体实施方式

[0011] 根据一个示例实施例，在分布式系统中可能存在大型图，其表示某个源代码存储库或某些源代码存储库的所有源代码。可以将源代码编制索引来创建图。在一些实施例中，该图可以存在于供开发者远程访问的一个机器或几个机器上，使得并不在各开发者的机器上运行该图，并且能够供多个开发者同时访问该图。在最简单的形式中，该图可以是两个表格，第一表格将简单名映射到合格名，并且第二表格映射合格名以构建系统依赖性。通过在分布式系统或云环境中创建和访问该图，在开发者之间能够缓存和重用计算。在其他实施例中，该图可以存在于开发者自己的机器上。

[0012] 如图2所示，示例源代码文件(201)中有两个简单名，行3上的“Bar”以及行5上的“Foo”。“Bar”的合格名在代码中被提供为行1上的“com.sourcecode.Bar”。然而，没有提供“Foo”的合格名。

[0013] 在一个示例系统中，在开发者机器上运行的集成开发环境(IDE)可以具有解析器(203)，其可以提供在特定源代码文件中使用的简单名的列表。解析器(203)也可以提供源代码文件中的合格名的列表。针对如图2所示的源代码文件，简单名的列表可以是：[Foo, Bar]。合格名的列表可以是：[com.sourcecode.Bar]。如图所示，可以不提供与简单名相对应的合格名。为了找到合格名以及为源代码中未提供对应合格名的简单名提供合格名的构建系统构造，可以针对简单名到合格名的可能扩展来查询源代码图(205)。在该给定的示例中，可以针对“Foo”的合格名以及与“Foo”相关联的构建依赖性来查询源代码图(205)。如果在该示例中提供合格名作为诸如“Bar”的导入，则构建系统依赖性可能存在或者可能不存在。因此，即使在源代码中提供完全合格名时，也可能针对构建系统依赖性来查询源代码图(205)。

[0014] 例如，当查找“Foo”时，对源代码图(205)的搜索可能匹配包含名“Foo”的类和构建系统构件。在一些实施例中，可以通过简单名在代码中的使用方式来过滤结果。例如，如果调用“Foo”的某个方法，则可以从潜在候选列表中滤除不包含该方法的补全的完全限定潜在候选。然后，查询可以返回导入和构建系统依赖性的列表。本领域普通技术人员能够领会，存在多种方式来排序和过滤该列表。

[0015] 可以遍历源代码图(205)以获得诸如在文件中找出的简单名的构建系统依赖性的信息。例如，源代码图可能能够为简单名“Foo”的合格名提供依赖性的目标，使得源代码能够被编译成目标文件。

[0016] 通过遍历源代码图(205)所获得的给定简单名的潜在合格名的列表可以用其他数据来充实并且进一步过滤(213)。充实数据可以包括:在存储库中使用合格名的次数(以允许排名);合格名是否被标记为弃用,并且如果是,则应使用何合格名作为替代;合格名包是否从当前正引用其的源代码可见(一些类可能被有意限制到某些软件包);以及该合格名可能为构建过程增添的复杂度,包括为构建源代码而增加的时间和/或资源。例如,可以基于已由其他源代码文件使用特定类型的合格名的次数来过滤列表。过滤还可以包括移除所有不可见的类或者移除在代码中的特定上下文内会不适合的所有类。例如,"Foo"可以调用源代码文件中的特定方法,因此不具有该方法的所有匹配的"Foo"库都可以从潜在候选列表中滤除。当向用户(215)返回补全(completion)和依赖性的可能候选的列表以供考虑时,示例系统可以基于已由其他开发者在源代码存储库的其他部分中使用特定合格名的频率来排名返回的结果。

[0017] 在一些实施例中,频繁使用的合格名是最有可能由开发者在他们的源代码文件中使用的名称。在其他排名当中,所述名也可以按照它们对它们在其中被视为使用的源代码文件是否可见、该名称表示的类是否具有在源代码文件中正使用的方法、以及这些类是否被弃用来排名。弃用的类可以在名称排名中降级。然后,可以向用户(215)示出用于代码中的特定简单名引用的潜在合格名候选的经充实、过滤的列表,并且针对简单命名的对象选择最适当的包、库或其他外部代码源。一旦用户(215)从合格名的候选中选择出源代码中的简单名的正确合格名,然后便可以将所选择的合格名再次发送到源图(205),以获取关于构建系统依赖性的信息,以便操控构建规则依赖性以及具体合格名的构建目标。在一些实施例中,用于补全和构建系统依赖性的潜在候选的列表可以被过滤,并且针对查询仅返回最有可能匹配的导入和构建系统依赖性。

[0018] 如图2所示,一旦在源代码图(205)中找出或者从文件本身(203)解析出唯一合格名(207、209)并且其与源代码文件(201)中使用的简单名匹配,便可以将合格名添加/导入到文件(201)中。可以周期性重新生成源图,以便确定最常使用哪些类/构建依赖性。

[0019] 如图3所示的示例方法开始于接收具有应当找出(或需要)其对应合格名的至少一个简单名的源代码文件(301)。然后,可以解析源代码文件以找出该文件中使用的简单名的列表以及该文件中的全部合格名的另一个列表(303)。针对应当找出对应合格名的至少一个的简单名,可以搜索全部源代码的图以找出该简单名的潜在候选合格名(305)。在一些实施例中,可以用附加信息过滤或者充实名称候选列表。可以将该列表提供给用户,以供该用户针对给定的简单名选择适当的合格名(309)。在一些实施例中,如果只有一个候选合格名,则用户可以不必选择合格名,并且可以自动选择该合格名。在其他实施例中,如果只有一个候选合格名,则用户可以仅确认该合格名是简单名的适当匹配。一旦选择合格名,一个示例方法便可以确定特定合格名的构建依赖性以及构建目标。可以将与源代码文件中的简单名匹配的合格名添加到源文件中。然后,可以重新生成源代码图,并且在将来的分析中可以使用关于简单名/合格名/构建依赖性频率的信息来找出合格名和构建依赖性。

[0020] 图4是布置用于托管和发布软件包的示例计算机(400)的高级框图。在非常基本的配置(401)中,计算设备(400)通常包括一个或多个处理器(410)和系统存储器(420)。存储器总线(430)能够被使用于处理器(410)与系统存储器(420)之间的通信。

[0021] 根据所需的配置,处理器(410)能够是任何类型,包括但不限于微处理器(μP)、微

控制器(μC)、数字信号处理器(DSP)或其任意组合。处理器(410)能够包括诸如一级缓存(411)和二级缓存(412)的一级或多级缓存、处理器内核(413)以及寄存器(414)。处理器内核(413)能够包括算术逻辑单元(ALU)、浮点单元(FPU)、数字信号处理内核(DSP内核)或其任意组合。存储器控制器(416)也能够与处理器(410)配用,或者在一些实施方式中,存储器控制器(415)能够是处理器(410)的内部部分。

[0022] 根据所需的配置,系统存储器(420)能够是任何类型,包括但不限于易失性存储器(诸如RAM)、非易失性存储器(诸如ROM、闪存等)或其任意组合。系统存储器(420)通常包括操作系统(421)、一个或多个应用(422)和程序数据(424)。应用(422)可以包括用于托管和发布软件包的方法。程序数据(424)包括存储的指令,所述指令在由一个或多个处理设备执行时实现用于托管和发布软件包的方法(423)。在一些实施例中,应用(422)能够被布置成与程序数据(424)一起在操作系统(421)上操作。

[0023] 计算设备(400)能够具有附加的特征或功能以及附加的接口,用以促进基本配置(401)与任何所需的设备和接口之间的通信。

[0024] 系统存储器(420)是计算机存储介质的示例。计算机存储介质包括但不限于RAM、ROM、EEPROM、闪速存储器或者其他存储器技术、CD-ROM、数字多功能盘(DVD)或者其他光学存储、磁带盒、磁带、磁盘存储或者其他磁存储设备、或者能够被用于存储所需信息并且能够由计算设备400来访问的任何其他介质。任何这样的计算机存储介质能够是设备(400)的一部分。

[0025] 计算设备(400)能够被实现为小型便携式(或移动式)电子设备的一部分,该电子设备诸如蜂窝电话、智能电话、个人数据助理(PDA)、个人媒体播放器设备、平板型计算机(平板)、无线web手表设备、个人头戴式送受话设备、专用设备或包括上述功能中任何功能的混合设备。计算设备(400)也能够被实现为包括膝上型计算机以及非膝上型计算机配置二者的个人计算机。

[0026] 前文的详细描述已经通过使用框图、流程图和/或示例来阐述设备和/或过程的各个实施例。就这样的框图、流程图和/或示例包含一个或多个功能和/或操作而言,本领域技术人员会理解到,这样的框图、流程图或示例中的每个功能和/或操作能够单独和/或共同地通过各种硬件、软件、固件或几乎其任意组合来实现。在一个实施例中,本文所述主题的几个部分可以经由专用集成电路(ASIC)、现场可编程门阵列(FPGA)、数字信号处理器(DSP)、其他集成格式或者作为web服务来实现。然而,本领域技术人员将认识到,本文中公开的实施例的一些方面能够全部或部分等同地以集成电路、作为在一个或多个计算机上运行的一个或多个计算机程序、作为在一个或多个处理器上运行的一个或多个程序、作为固件或者作为几乎其任意组合来实现,并且鉴于本公开,设计电路和/或编写软件和/或固件的代码会完全在本领域技术人员的技术范围内。此外,本领域技术人员将领会到,本文所述主题的机制能够作为各种形式的程序产品分发,并且本文所述主题的说明性实施例无论用于实际执行该分发的非暂时性信号承载介质的特定类型如何都适用。非暂时性信号承载介质的示例包括但不限于以下各项:可记录型介质,诸如软盘、硬盘驱动器、压缩盘(CD)、数字视频盘(DVD)、数字磁带、计算机存储器等;以及传输介质,诸如数字和/或模拟通信介质(例如,光纤电缆、波导、有线通信链路、无线通信链路等)。

[0027] 就本文中大体上任何复数和/或单数术语的使用而言,本领域技术人员能够视场

境(context)和/或应用将复数转为单数和/或将单数转为复数。为清楚起见,本文可以明确阐述各个单数/复数置换。

[0028] 因此,已对所述主题的特定实施例予以描述。其他实施例在所附权利要求书的范围内。在一些情况下,能够以不同的次序来执行权利要求中所记载的动作并且仍能获得期望的结果。此外,在附图中所描绘的过程不一定要求所示的特定次序或者顺序次序来获得期望的结果。在某些实施方式中,多任务以及并行处理可能是有利的。

```
1. import java.util.List;
2. public class classDemo{
3. List theList = new ArrayList();
4. ....
}
```

图1

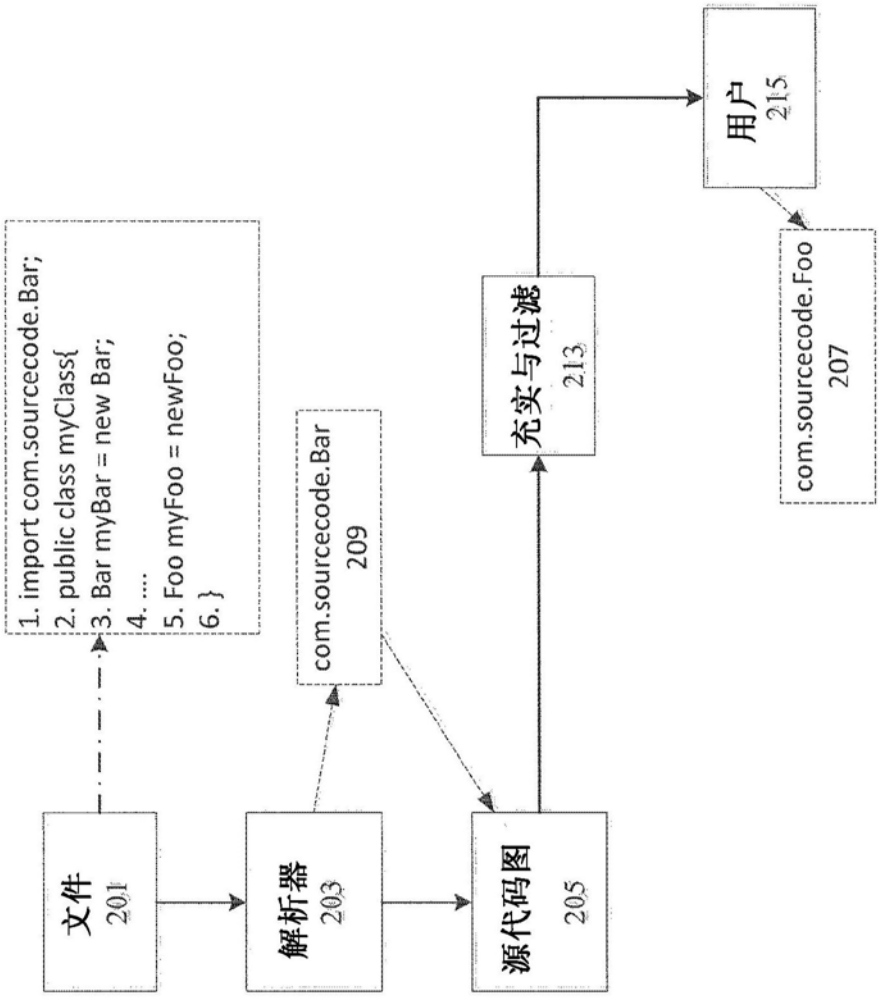


图2

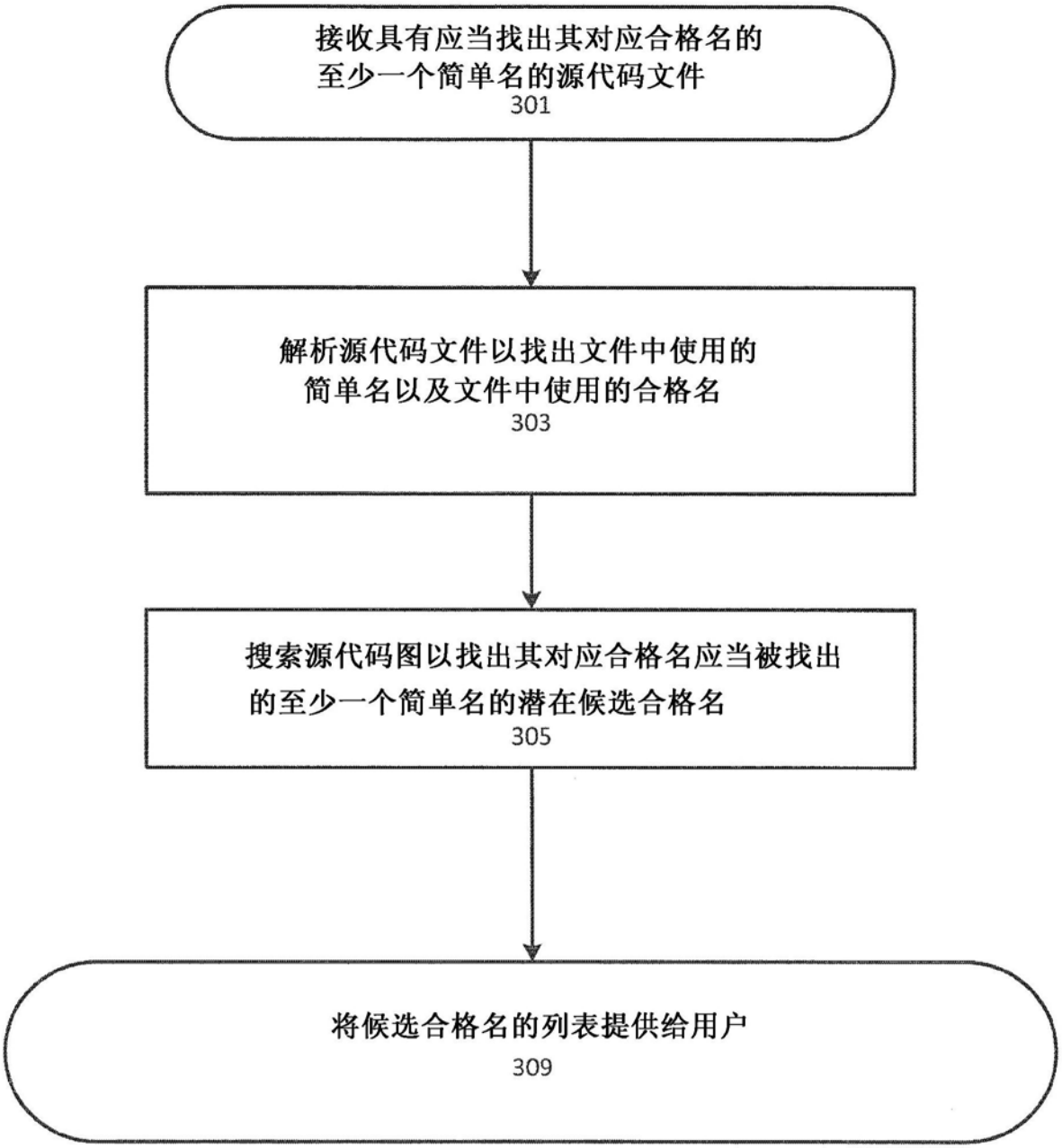

接收具有应当找出其对应合格名的
至少一个简单名的源代码文件
301
解析源代码文件以找出文件中使用的
简单名以及文件中使用的合格名
303
搜索源代码图以找出其对应合格名应当被找出
的至少一个简单名的潜在候选合格名
305
将候选合格名的列表提供给用户
309

图3

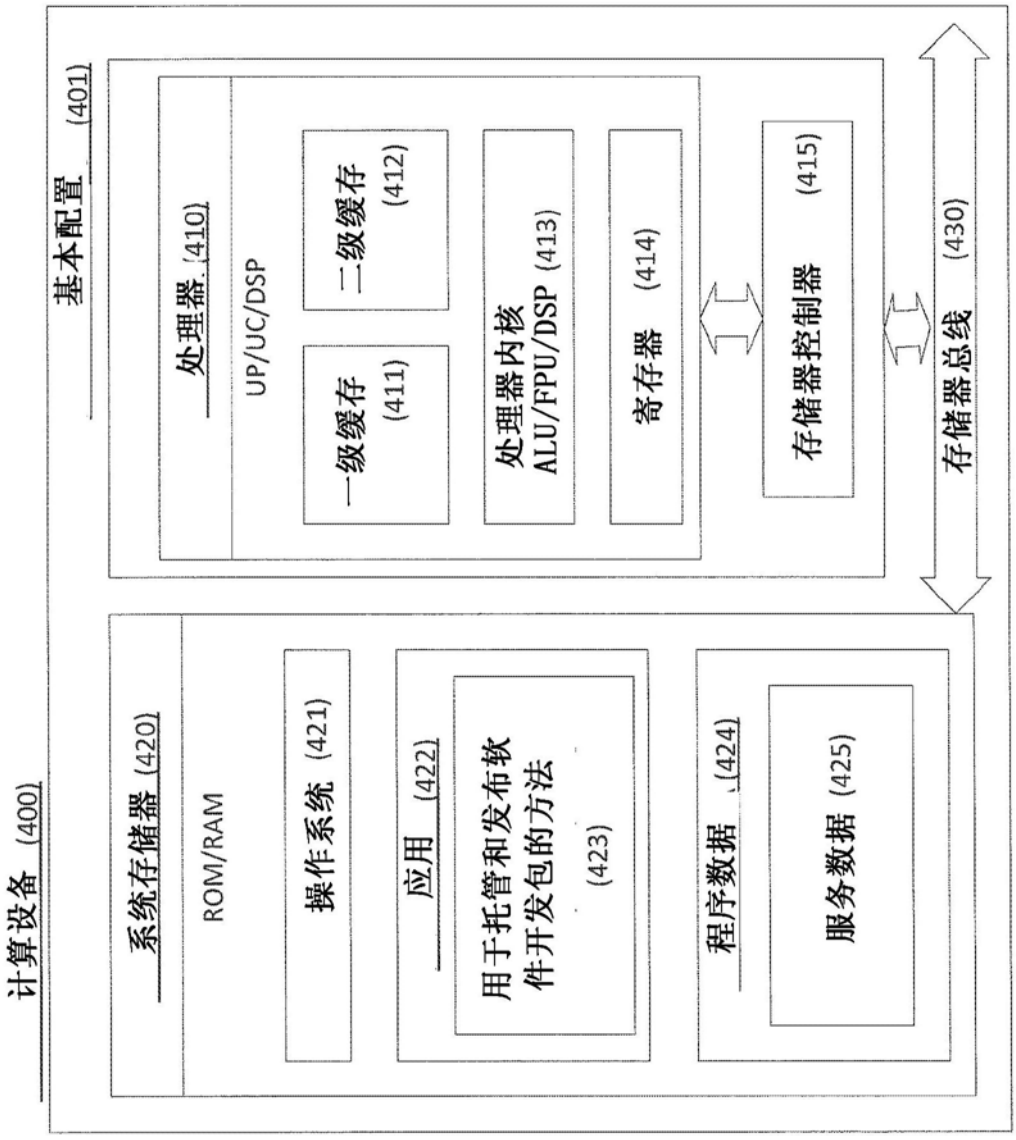

计算设备 (400)
基本配置 (401)
处理器 (410)
UP/UC/DSP
一级缓存 (411)
二级缓存 (412)
处理器内核 ALU/FPU/DSP (413)
寄存器 (414)
存储器控制器 (415)
存储器总线 (430)
系统存储器 (420)
ROM/RAM
操作系统 (421)
应用 (422)
用于托管和发布软件开发包的方法 (423)
程序数据 (424)
服务数据 (425)

图4