(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 106537333 A
(43)申请公布日 2017.03.22

(21)申请号 201580031457.1

(22)申请日 2015.06.10

(30)优先权数据
62/012,127 2014.06.13 US

(85)PCT国际申请进入国家阶段日
2016.12.12

(86)PCT国际申请的申请数据
PCT/US2015/035148 2015.06.10

(87)PCT国际申请的公布数据
WO2015/191746 EN 2015.12.17

(71)申请人 查尔斯斯塔克德拉珀实验室公司
地址 美国马萨诸塞州

(72)发明人 R·T·卡巴克三世 B·D·加伊诺 N·A·布洛克 E·T·安特尔曼

(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华 庞淑敏

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

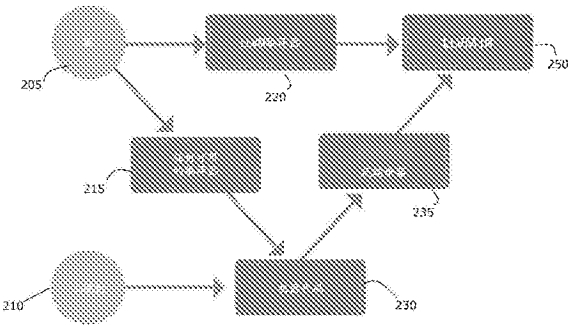
权利要求书3页 说明书17页 附图9页

(54)发明名称

用于软件产物的数据库的系统和方法

(57)摘要

示出了用于提供语料库的系统、方法以及计算机程序产品。示例性实施方式包括自动地获得多个软件文件，针对所述多个软件文件中的每一个确定多个产物，并且将针对所述软件文件中的每一个的所述多个产物存储在数据库中。附加实施方式通过将所述软件文件中的每一个转换成中间表示并根据用于所述软件文件中的每一个的所述中间表示，来确定所述产物中的至少一些而确定用于所述多个软件文件中的每一个的一些产物。特定示例性实施方式通过从所述多个软件文件中的每一个提取一串字符串来确定用于软件文件的每一个的产物中的至少一些。所述软件文件可以是源代码或二进制格式。



CN 106537333 A

1.一种用于提供语料库的方法,包括:

获得多个软件文件;

针对所述多个软件文件中的每一个确定多个产物;以及

将针对所述多个软件文件中的每一个的所述多个产物存储在数据库中。

2.根据权利要求1所述的方法,还包括:定位所述多个软件文件中的构建文件并使用所述构建文件来生成编译器调用。

3.根据权利要求2所述的方法,还包括:将所述编译器调用转换成底层虚拟机(LLVM)前端调用。

4.根据权利要求3所述的方法,其中所述LLVM前端调用被修改或设备化以生成产物。

5.根据权利要求2所述的方法,其中所述构建文件选自包括autocomf文件、cmake文件、automake文件、make文件以及供应商指令的组。

6.根据权利要求2所述的方法,其中使用所述构建文件来生成所述编译器调用包括尝试使用所述构建文件来进行至少部分地完成的构建。

7.根据权利要求2所述的方法,其中使用所述构建文件包括自动地使用所述构建文件。

8.根据权利要求2所述的方法,其中通过使用系统调用挂钩来识别原始构建过程中的一个或多个构建步骤并将其设备化,来确定生成的所述编译器调用。

9.根据权利要求8所述的方法,其中所述系统调用挂钩包括s轨迹挂钩。

10.根据权利要求1所述的方法,其中获得多个软件文件包括自动地获得多个软件文件。

11.根据权利要求10所述的方法,其中自动地获得多个软件文件包括使多个计算机共同地获得所述多个软件文件。

12.根据权利要求10所述的方法,其中自动地获得多个软件文件包括从公共储存库自动地获得所述多个软件文件中的至少一些。

13.根据权利要求10所述的方法,其中所述多个软件文件包括软件包的至少一个修订版。

14.根据权利要求13所述的方法,还包括在所述软件包的所述至少一个修订版的产物之间的多个关系,其中所述多个关系被存储在所述数据库中。

15.根据权利要求1所述的方法,还包括:将所述软件文件中的每一个转换成中间表示,并根据针对所述软件文件中的每一个的所述中间表示来确定所述多个产物中的至少一个。

16.根据权利要求1所述的方法,还包括:将所述多个软件文件中的一个或多个分布在多个计算机之间并使所述多个计算机共同地将所述软件文件中的每一个转换成中间表示,并且根据针对所述软件文件中的每一个的所述中间表示来确定所述多个产物中的至少一个。

17.根据权利要求1所述的方法,其中所述多个产物包括调用图、控制流程图、use-def链、def-use链、支配树、基本块、变量、常量、分支语义以及协议中的一个或多个。

18.根据权利要求1所述的方法,其中所述多个产物包括系统调用轨迹和执行轨迹中的一个或多个。

19.根据权利要求1所述的方法,其中所述多个产物包括循环不变量、类型信息、Z符号以及标签迁移系统表示中的一个或多个。

20.根据权利要求1所述的方法，其中所述多个产物包括内联代码注释、提交历史、文件编制文件以及公共漏洞和暴露源入口中的一个或多个。

21.根据权利要求1所述的方法，其中针对所述多个软件文件中的每一个确定所述多个产物包括通过从所述多个软件文件中的至少一个提取字符串来确定所述多个产物中的至少一个。

22.根据权利要求1所述的方法，其中针对所述多个软件文件中的每一个确定所述多个产物包括在设备化环境中运行所述多个软件文件中的至少一些。

23.根据权利要求22所述的方法，其中所述设备化环境选自包括虚拟机、模拟器以及系统管理程序的组。

24.根据权利要求1所述的方法，还包括：生成与用于所述软件文件中的每一个的所述多个产物相关联的多个分级关系。

25.根据权利要求1所述的方法，还包括：将所述多个软件文件存储在所述数据库中。

26.根据权利要求1所述的方法，其中所述多个软件文件是源代码格式。

27.根据权利要求1所述的方法，其中所述多个软件文件是二进制代码格式。

28.根据权利要求1所述的方法，其中所述数据库是图形数据库。

29.一种用于提供语料库的装置，包括：

一个或多个存储设备，存储针对多个软件文件中的每一个的多个产物，其中所述多个产物中的至少一些是根据所述多个软件文件中的至少一些的中间表示而确定的。

30.一种用于提供语料库的系统，包括：

接口，能够与具有多个软件文件的源进行通信；

一个或多个存储设备，用于存储针对所述多个软件文件中的每一个的多个产物；以及

处理器，被通信耦合到所述接口和所述存储设备，并且被配置成：

从所述源获得所述多个软件文件，以及

针对所述多个软件文件中的每一个确定所述多个产物。

31.根据权利要求30所述的系统，其中所述接口是网络接口。

32.根据权利要求30所述的系统，其中所述处理器被配置成确定所述多个产物包括：所述处理器被配置成将所述软件文件中的每一个转换成中间表示并根据针对所述软件文件中的每一个的所述中间表示来确定所述多个产物中的至少一个。

33.根据权利要求30所述的系统，其中所述处理器被配置成确定所述多个产物包括：所述处理器被配置成通过从所述多个软件文件中的至少一些提取字符串来确定所述多个产物中的至少一个。

34.根据权利要求30所述的系统，其中所述处理器被配置成获得所述多个软件文件包括：所述处理器被配置成自动地从软件储存库检索所述多个软件文件。

35.根据权利要求30所述的系统，其中所述多个产物包括针对所述多个软件文件中的每一个的图形产物。

36.根据权利要求30所述的系统，其中所述多个产物包括针对所述多个软件文件中的每一个的开发产物。

37.根据权利要求30所述的系统，其中所述多个产物包括针对所述多个软件文件中的每一个的动态产物。

38.根据权利要求30所述的系统，其中所述多个产物包括针对所述多个软件文件中的每一个的导出产物。

39.一种其上存储有可执行程序的非瞬态计算机可读介质，其中，所述程序命令处理设备执行以下步骤：

自动地获得多个软件文件；

通过以下操作来针对所述多个软件文件中的每一个确定多个产物：

将所述软件文件中的每一个转换成中间表示，并且

根据针对所述软件文件中的每一个的所述中间表示来确定所述多个产物中的至少一个；以及

将针对所述多个软件文件中的每一个的所述多个产物存储在数据库中。

用于软件产物的数据库的系统和方法

[0001] 相关申请

[0002] 本申请要求2014年6月13日提交的美国临时申请No.62/012,127的权益。上述申请的全部教导内容通过引用而被整体结合到本文中。

[0003] 政府支持

[0004] 本发明是根据来自美国空军的许可号FA8750-14-C-0056和来自国防高级研究项目管理局的许可号FA8750-15-C-0242在政府支持下完成的。政府在本发明中具有一些权益。

背景技术

[0005] 现在，软件开发、维护以及修复是手动过程。软件供应商随时间推移而规划、实现、文件编制、测试、部署和维护计算机程序。初始的规划、实现、文件编制、测试和部署常常是不完整的，并且总是缺少期望的特征或包含缺陷。许多供应商使用生命周期维护计划以通过随着软件的成熟推送迭代修正版、安全补丁以及特征增强来解决这些缺点。

[0006] 在全世界数十亿的线路中部署了大量的软件代码，并且花费大量的时间和金钱来解决维护和修正版。历史上，软件维护是自组织且是反应性(即，对错误报告、安全漏洞报告以及用户针对特征增强的请求进行响应)的手动过程。

发明内容

[0007] 本发明的实施方式有助于使软件开发、维护以及修复生命周期的关键方面自动化，包括例如找到程序缺陷，诸如错误(代码中的错误)、安全漏洞以及协议缺陷。本发明的示例性实施方式提供了可以利用大量软件文件(包括公开可用的那些软件文件或专用软件)的系统和方法。

[0008] 示例性实施方式中的特定实施方式可以自动地识别用于软件文件的最新版本或补丁。附加实施方式可以自动地对已知存在于特定软件文件中的设计模式(诸如软件缺陷(例如，错误、漏洞、协议缺陷)和修复进行定位。其它实施方式可以通过在软件文件(对于该软件文件，先前并不知道该文件包含缺陷)中对已知缺陷进行定位来利用该已知缺陷。附加实施方式可以自动地定位设计模式，诸如识别源或二进制代码的各部分，以识别文件、程序、函数或代码块。

[0009] 根据本发明的一个实施方式，一种用于提供语料库(corpus)的示例性方法包括：获得多个软件文件，针对所述软件文件的每一个确定多个产物(artifacts)，并将针对所述软件文件的每一个的产物存储在数据库中。附加实施方式通过将软件文件的每一个转换成中间表示并根据用于软件文件的每一个的中间表示来确定产物中的至少一个，而确定针对软件文件的每一个的一些产物。一些示例性实施方式通过从所述多个软件文件中的至少一些提取字符串，来确定用于软件文件的每一个的产物中的至少一些。

[0010] 附加实施方式还可以自动地获得软件文件，包括通过使多个计算机共同地获得软件文件，诸如从公共软件储存库获取。附加实施方式可以在所述多个软件文件中定位构建

(build)文件(诸如autocomf文件、cmake文件、automake文件、make文件以及供应商指令)并使用该构建文件来生成编译器调用。特定实施方式可以通过首先使用系统调用挂钩从原始构建过程获得构建步骤而生成编译器调用。系统调用挂钩是可以拦截(也称为钩住)调用、消息或事件(包括拦截操作系统调用或在软件组件之间传递的调用)的代码。附加实施方式还可以将编译器调用转换成底层虚拟机(LLVM)前端调用。针对特定实施方式,转换编译器调用包括执行钩住,诸如s轨迹钩住。针对特定实施方式,可以对LLVM前端调用进行修改或设备化以产生产物。针对一些实施方式,使用构建文件来生成编译器调用包括尝试使用构建文件来进行至少部分地完成的构建,其是进行编译但并未适当地链接的构建文件。针对特定实施方式,使用构建文件是自动地使用的。针对特定示例性实施方式,所述多个产物可以包括静态产物、动态产物、导出产物和/或元数据产物。针对特定示例性实施方式,所述多个产物可以包括图形产物和/或开发产物。针对特定实施方式,所述多个软件文件包括软件包的至少一个修订版,其是文件与关于那些文件的信息的组合件。特定附加实施方式还包括在软件包的修订版的产物中的至少一些之间的多个关系,并且该关系被存储在数据库中。

[0011] 附加示例性实施方式还可以将软件文件中的一个或多个分布在多个计算机之间,并且使计算机共同地将软件文件的每一个转换成中间表示且根据用于软件文件的每一个的中间表示来确定产物中的至少一个。其它附加实施方式还可以生成用于软件文件的每一个的产物并将其布置成分级交互关系。特定示例性实施方式还可以将软件文件存储在数据库中。

[0012] 针对本发明的一些附加实施方式,确定用于软件文件的每一个的产物包括在设备化环境(诸如虚拟机、模拟器或系统管理程序)中运行软件文件。此特征允许确定多种附加产物,并且可以支持许多操作系统。

[0013] 针对特定示例性实施方式,产物可以包括调用图、控制流程图、use-def链、def-use链、支配树、基本块、变量、常量、分支语义以及协议。针对特定示例性实施方式,产物可以包括系统调用轨迹和执行轨迹。针对特定示例性实施方式,产物可以包括循环不变量、类型信息、Z符号(Z notation)以及标签迁移系统表示。针对一些示例性实施方式,产物可以包括内联代码注释、提交历史、文件编制文件以及公共漏洞和暴露源入口。示例性方法的特定附加实施方式还可以自动地从软件储存库检索软件文件。针对特定示例性实施方式,软件文件采取源代码格式或二进制代码格式。

[0014] 本发明的附加示例性实施方式是一种用于提供数据库语料库的装置。示例性装置是可以存储用于软件文件的产物的一个或多个存储设备,其中可以根据软件文件的中间表示来确定所述产物中的至少一个。

[0015] 附加示例性实施方式是一种用于提供语料库的系统,该系统包括:接口,其能够与具有多个软件文件的源进行通信;一个或多个存储设备,其用于存储用于软件文件的每一个的产物;以及处理器,该处理器被通信耦合到所述接口和所述存储设备,并被配置成:从所述源获得所述多个软件文件,并针对所述软件文件的每一个确定产物。针对特定实施方式,可以自动地获得所述文件,并且可以自动地完成确定产物。

[0016] 针对示例性系统的特定实施方式,所述接口可以是网络接口。针对特定示例性实施方式,所述处理器还被配置成通过将软件文件的每一个转换成中间表示并根据用于软件

文件的每一个的中间表示来确定一些产物，来确定所述产物中的一些。针对特定示例性实施方式，所述处理器还被配置成通过从软件文件中的至少一些提取一串字符来确定所述产物中的一些。针对附加示例性实施方式，所述处理器被配置成自动地从软件储存库检索软件文件。

[0017] 本发明的另一示例性实施方式是其上存储有可执行程序的非瞬态计算机可读介质，其中所述程序命令处理设备执行以下步骤：自动地获得软件文件；通过以下操作来针对软件文件的每一个确定产物：(i)将软件文件的每一个转换成中间表示，(ii)根据用于软件文件的每一个的中间表示来确定一些产物，以及(iii)通过从软件文件中的至少一些提取一串字符来确定一些产物；以及，将用于软件文件的每一个的所述多个产物存储在数据库中。

附图说明

[0018] 根据如附图中所示的本发明的示例性实施方式的以下更特定描述，前述内容将是显见的，在附图中相同的附图标记贯穿不同视图指代相同部分。附图不一定按比例，而是着重于图示本发明的实施方式。

[0019] 图1是图示出用于提供用于软件文件的语料库的方法的示例性实施方式的流程图。

[0020] 图2是图示出根据本发明的实施方式的用以从用于语料库的输入软件文件提取中间表示(IR)的示例性处理的流程图。

[0021] 图3是图示出根据本发明的实施方式的用于软件文件的产物之间的分级关系的框图。

[0022] 图4是图示出用于提供用于软件文件的产物的语料库的系统的示例性实施方式的框图。

[0023] 图5是图示出用于识别设计模式的方法的示例性实施方式的框图。

[0024] 图6是图示出用于识别缺陷的方法的示例性实施方式的流程图。

[0025] 图7是图示出根据本发明的实施方式的用于识别设计模式的产物的聚类的框图。

[0026] 图8是图示出用于使用语料库来识别软件文件的方法的示例性实施方式的流程图。

[0027] 图9是图示出用于识别程序片段的方法的示例性实施方式的流程图。

[0028] 图10是图示出根据本发明的实施方式的使用所述语料库的系统的框图。

具体实施方式

[0029] 下面是本发明的示例性实施方式的描述。在本文中引用的任何专利或公开的全部教导内容被通过引用结合到本文中。

[0030] 根据本公开的示例性实施方式的软件分析允许利用来自现有软件文件的知识，所述现有文件包括来自公开可用源或作为专用软件的文件。此知识然后可以应用于其它软件文件，包括修复缺陷、识别漏洞、识别协议缺陷或建议代码改进。

[0031] 本发明的示例性实施方式可以针对软件分析的变化方面，包括创建、更新、维护或者以其他方式提供软件文件的语料库和用于知识数据库的关于软件文件的相关产物。根据

本发明的各方面，此语料库可以被用于多种目的，包括自动地识别软件文件的更新版本、可用于软件文件的补丁、已知具有缺陷的文件中的这些缺陷以及先前未被已知包含这些错误的文件中的已知缺陷。本发明的实施方式还利用来自语料库的知识来解决这些问题。

[0032] 图1是图示出根据本发明的实施方式的用于语料库的输入软件文件的示例性处理的流程图。首先示出的步骤是获得多个软件文件110。这些软件文件可以采取源代码格式(其通常是纯文本)，或者采取二进制代码格式或一些其它格式。此外，针对本发明的特定示例性实施方式，源代码格式可以是可被编译的任何计算机语言，包括Ada、C/C++、D、Erlang、Haskell、Java、Lua、Objective C/C++、PHP、Pure、Python以及Ruby。针对特定附加示例性实施方式，还可以获得解释语言以供本发明的实施方式使用，包括PERL和bash脚本。

[0033] 获得的软件文件不仅包括源代码或二进制文件，而且可以包括与那些文件或相应软件项目相关联的任何文件。例如，软件文件还包括关联构建文件、make文件、库、文件编制文件、提交日志、修订历史、bugzilla入口、公共漏洞和暴露(CVE)条目及其它非结构化文本。

[0034] 可以从各种源获得软件文件。例如，可以通过网络接口经由因特网从诸如GitHUB、SourceForge、BitBucket、GoogleCode或公共漏洞和暴露系统之类的公开可用软件储存库(诸如由MITRE公司维护的软件储存库)来获得软件文件。一般地，这些储存库包含文件和对该文件所进行的改变的历史。并且，例如，可以提供统一资源定位符(URL)以指向可以从其获得文件的站点。还可以经由接口从私有网络获得或者从本地硬盘驱动器或其它存储设备本地获得软件文件。该接口提供到源的通信耦合。

[0035] 本发明的示例性实施方式可以获得从源可获得的一些、大多数或所有文件。此外，一些示例性实施方式还使获得文件自动化，并且例如可以自动地下载文件、整个软件项目(例如，修订历史、提交日志、源代码)、项目或程序的所有修订版、目录中的所有文件或从源可获得的所有文件。一些实施方式通过针对整个储存库的每个修订版进行爬取来获得所有可用软件文件。特定示例性实施方式获得用于语料库中的每个软件项目的整个源控制储存库，以促进自动地获得用于该项目的所有关联文件，包括获得每个软件文件修订版。用于储存库的示例性源控制系统包括Git、Mercurial、Subversion、Concurrent Versions System(并发版本系统)、BitKeeper以及Perforce。特定实施方式还可以连续地或周期性地核对源，以辨别该源是否已被改变或更新，并且如果是这样，则可以仅仅从该源获得该改变或更新，或者还再次获得所有软件文件。许多源具有确定对源的改变的方法，诸如示例性实施方式在从源获得更新时可以使用的添加日期或改变日期字段。

[0036] 本发明的特定示例性实施方式还可以分别地获得库软件文件，所述库软件文件可以在储存库不包含这些库的情况下由从储存库获得的源代码文件用来解决对此类文件的需要。这些实施方式中的特定实施方式尝试获得合理地从任何公开源可用的或者从软件供应商获得的任何库软件文件，以便包括在语料库中。另外，特定实施方式允许用户提供由软件文件使用的库或者识别所使用的库，以使得可以获得这些库。一些实施方式抓取用于每个项目的软件文件，以识别由该项目使用的库，使得这些库可以被获得且也被安装，如果需要的话。

[0037] 根据本发明的示例性方法中的下一步骤是针对所述多个软件文件120中的每一个确定多个产物(artifacts)。软件产物可以描述软件文件的功能、架构或设计。产物类型的

示例包括静态产物、动态产物、导出产物以及元数据产物。

[0038] 示例性方法的最后一个步骤是将针对所述多个软件文件中的每一个的所述多个产物存储在数据库130中。所述多个产物以这样的方式存储，该方式使得这多个产物可以被识别为对应于根据其来确定多个产物的特定软件文件。此识别可以以众所周知的各种方式中的任何一个来完成，诸如入数据库架构(schema)所表示的数据库中的字段、指针、所存储的位置或者任何其它标识符，诸如文件名。属于同一项目或构建的文件可以被同样地跟踪，以使得可以保持关系。

[0039] 针对不同的实施方式，数据库可以采取不同的形式，诸如图形数据库、关系数据库或平面文件。一个优选实施方式采用OrientDB，其是由Orient Technologies所领导的OrientDB Open Source Project(开源项目)提供的分布式图形数据库。另一优选实施方式采用Titan(其是针对存储并查询跨多机器聚类分布的图形而被优化的可缩放图形数据库)以及Apache Cassandra存储后端。特定示例性实施方式还可以采用来自Paradigm4的SciDB，其是也存储图形产物并对其进行操作的阵列数据库。

[0040] 一般地可以从源代码文件、二进制文件或其它产物确定静态产物、动态产物、导出产物以及元数据产物。下面提供了这些类型的产物的示例。示例性实施方式可以针对源代码或二进制软件文件确定这些产物中的一个或多个。特定实施方式并不确定这些产物类型中的每一个或者用于特定类型的每个产物，而是替代地，可以确定产物类型的子集和/或一个类型内的产物的子集，和/或根本不确定特定类型。

[0041] 静态产物(static artifacts)

[0042] 用于软件文件的静态产物包括调用图、控制流程图、use-def链、def-use链、支配树、基本块、变量、常量、分支语义以及协议。

[0043] 调用图(CG)是被一函数调用的各个函数的有向图。GG表示高级程序结构，并且被描绘为节点，图中的每个节点均表示函数，并且节点之间的每个边是有向的且显示一个函数是否可以调用另一函数。

[0044] 控制流图(CFG)是函数内部的基本块之间的控制流的有向图。CFG表示函数级程序结构。CFG中的每个节点表示基本块，并且节点之间的边是有向的并示出流中的潜在路径。

[0045] Use-Def(UD)和Def-Use链(DU)是在代码的基本块中执行的输入(使用)、输出(定义)以及操作的无环有向图。例如，UD链是变量的使用和在不插入重定义的情况下可以到达该使用的该变量的所有定义。DU链是变量的定义和在不插入重定义的情况下从该定义所能到达的所有使用。这些链使得能够关于被接受的输入类型、所生成的输出类型以及在代码的基本块内部执行的操作，来进行代码的基本块的语义分析。

[0046] 支配树(DT)是表示CGF中哪些节点支配其它节点(在其路径中)的矩阵。例如，如果从入口节点到第二节点的每个路径必须经历第一节点，则第一节点支配第二节点。以Pre(从入口向前)和Post(从出口向后)形式来表示DT。当路径改变到CGF中的特定节点时，DT突出显示。

[0047] 基本块是CGF的每个节点内部的指令和操作数。可以比较基本块，并且可以产生两个基本块之间的相似性度量。

[0048] 变量(Variable)是用于信息及其类型的存储单位，表示针对任何函数参数、局部变量或全局变量其可以存储的信息的类型，并且如果有默认值可用的话还包括默认值。它

们可以提供关于程序的初始状态和基本约束，并且示出类型或初始值方面的变化，其可以影响程序行为。

[0049] 常量(Constants)是任何常量的类型和值，并且可以提供关于程序的初始状态和基本约束。它们可以示出类型或初始值的变化，其可以影响程序行为。

[0050] 分支语义(Branch Semantics)是if语句和循环内部的布尔值评估。分支控制基本块被执行的条件。

[0051] 协议(Protocols)是协议、库、系统调用以及程序所使用的其它已知函数的名称和引用。

[0052] 本发明的示例性实施方式可以自动地从诸如由公开可用的LLVM(前面的底层虚拟机)编译器基础设施项目所提供的软件源代码文件的中间表示(IR)来确定静态产物。LLVM IR是底层公共语言，其可以有效地表示高级语言且独立于指令集架构(ISA)，诸如ARM、X86、X64、MIPS以及PPC。可以使用用于不同计算机语言的不同LLVM编译器(也称为前端)来将源代码变换成公共LLVM IR。用于至少Ada、C/C++、D、Erlang、Haskell、Java、Lua、Objective C/C++、PHP、Pure、Python以及Ruby的前端是公开可用的。此外，用于附加语言的前端可以被容易地编程。LLVM还具有可用的优化器和后端，该后端可以将LLVM IR变换成用于多种不同ISA的机器语言。附加示例性实施方式可以从源代码文件确定静态产物。

[0053] 图2是图示出根据本发明的实施方式的可以被利用的用于语料库的输入软件文件的附加示例性处理的流程图。除其它的之外，示例性实施方式还可以获得源代码205和二进制代码210软件文件两者。当LLVM编译器220可用于源代码文件205的语言时，可以使用用于该语言的LLVM编译器220来将源代码翻译成LLVM IR 250。针对没有可用LLVM编译器的编译语言，首先可以利用针对该语言的任何支持的编译器215将源代码205编译成二进制文件230。然后，使用诸如Fracture之类的解编译器235(其是由Draper Laboratory提供的公开可用开源解编译器)将二进制文件230解编译。解编译器235将机器代码230翻译成LLVM IR 250。针对以二进制形式210获得的文件，该形式是机器代码230，使用解编译器235将其解编译以获得LLVM IR 250。示例性实施方式可以从LLVM IR提取语言无关且ISA无关的产物。

[0054] 本发明的示例性实施方式可以自动地获得用于每个源代码软件文件的IR。例如，示例性实施方式可以自动地在储存库中搜索用于标准构建文件(诸如autocomf、cmake、automake或make文件)或供应商指令的项目。示例性实施方式可以通过监视构建过程并将编译器调用转换成用于源代码的特定语言的LLVM前端调用，来自动地选择性地尝试使用此类文件来构建项目。用于构建文件的选择过程可以逐步通过每个文件以确定哪些存在并提供已完成产物或部分完成产物。

[0055] 附加示例性实施方式可以在自动地从储存库获得文件、将文件转换成LLVM IR和/或针对文件确定产物时，使用分布式计算机系统。示例性分布式系统可以使用主计算机来向外推送项目和构建至从机器以进行处理。从设备每个可以处理其被分配的项目、版本、修订版或构建，并且可以将源或二进制文件翻译成LLVM IR和/或确定产物并提供结果以便存储在语料库中。一些示例性实施方式可以采用Hadoop，其是用于非常大的数据集的分布式存储和分布式处理的开源软件框架。还可以将从源储存库获得文件分布在一组机器之间。

[0056] 根据示例性实施方式还可以将软件文件和LLVM IR存储在语料库中，包括用分布式存储库。示例性实施方式还可以确定软件文件或LLVM IR代码已被存储在数据库中并选

择不再次存储文件。可以使用指针、图形数据库中的边或其它引用标识符来将文件与特定项目、目录或文件的其它集合相关联。

[0057] 动态产物

[0058] 动态产物表示程序行为(behavior)，并且是通过在设备化环境(诸如虚拟机、模拟器(例如快速模拟器(“QEMU”))或系统管理程序)中运行软件而生成的。动态产物包括系统调用轨迹/库轨迹以及执行轨迹。

[0059] 系统调用轨迹或库轨迹是系统调用或库调用被执行的顺序和频率。系统调用是程序如何从管理输入/输出请求的操作系统的内核请求服务。库调用是对软件库的调用，该软件库是可以被重新用来开发软件程序和应用程序的编程代码的集合。

[0060] 执行轨迹是每个指令轨迹，其包括指令字节、栈框架、存储器使用(例如，驻留/工作组尺寸)、用户/内核时间及其它运行时信息。

[0061] 本发明的示例性实施方式可以产生(spawn)虚拟环境，包括针对各种操作系统的虚拟环境，并且可以运行并编译源代码和二进制文件。这些环境可以允许动态产物被确定。例如，可以采用诸如Valgrind或Daikon之类的公开可用程序来提供关于程序的运行时信息以充当产物。Valgrind是用于(除其它的之外)调试存储器、检测存储器泄漏以及行为评测的工具。Daikon是可以检测代码中的不变量的程序；不变量是在代码中的一些点处保持为真的条件。

[0062] 其它实施方式可以采用附加诊断和调试程序或实用工具，诸如strace和dtrace，其是公开可用的。strace被用来监视进程与内核之间的交互，包括系统调用。dtrace可以用来为系统提供运行时信息，包括所使用的存储器量、CPU时间、特定函数调用以及访问特定文件的进程。示例性实施方式还可以跨程序的多次运行而跟踪执行轨迹(例如，使用Valgrind)。

[0063] 附加实施方式可以通过KLEE引擎来运行LLVM IR。KLEE是符号虚拟机，其是公开可用的开源代码。KLEE以符号方式执行LLVM IR并自动地生成训练所有代码程序路径的测试。符号执行涉及(除其它的之外)分析代码以确定什么输入促使代码的每个部分执行。利用KLEE在发现函数正确性误差和行为不一致性时是非常有效的，并且因此允许本发明的示例性实施方式快速地识别类似代码的差异(例如，跨各修订)。

[0064] 导出产物

[0065] 导出产物表示复杂的高级程序行为，并且提取作为这些行为的特性(characteristic)的属性和事实。导出产物包括程序特性、循环不变量、扩展类型信息、Z符号和标签迁移系统表示。

[0066] 程序特性是关于从执行轨迹导出的程序的事实。这些事实包括最小、最大以及平均存储器尺寸；执行时间；以及栈深度。

[0067] 循环不变量是在循环的所有迭代(或所选的一组迭代)内被保持的属性。循环不变量可以被映射到分支语义以揭示类似行为。

[0068] 扩展类型信息包括关于类型的事实，包括变量可以拥有的值的范围、与其它变量的关系以及可以被抽象化的其它特征。类型约束可以显示关于该代码的行为和特征。

[0069] Z符号基于Zermelo-Fraenkel集合理论。其提供分类型的代数标记法，使得能够实现在基本块与整个函数之间的忽视结构、顺序以及类型的比较度量。

[0070] 标签迁移系统(LTS)表示是表示从程序抽象化的高级状态的图形系统。图的节点是状态,并且各边用迁移中的关联动作来标记。

[0071] 针对特定示例性实施方式,可以根据其它产物、根据源代码文件(包括使用上文针对动态产物所述的程序)以及根据LLVM IR来确定导出产物。

[0072] 元数据产物

[0073] 元数据产物表示程序上下文,并且包括与代码相关联的元数据。这些产物具有与计算机程序的上下文关系。元数据产物包括文件名、修订号、文件的时间戳、哈希值以及文件的位置,诸如属于特定目录或项目。可以将元数据产物的子集称为开发产物,其是涉及文件、程序或项目的开放过程的产物。开发产物可以包括内联代码注释、提交历史、bugzilla入口、CVE入口、构建信息、配置脚本以及文件编制文件,诸如README.*TODO.*。

[0074] 示例性实施方式可以采用Doxygen,其是公开可用的文件编制生成器。Doxygen可以从特殊注释的源代码文件生成用于程序员和/或最终用户的软件文件编制(即内联代码文件编制)。

[0075] 附加实施方式可以采用解析器(诸如另一语言识别工具(ANTLR)4生成解析器)来产生抽象语法树(AST),以提取也可以充当产物的高级语言特征。ANTLR4针对用于语言的串采用语法产生规则,并且生成可以构建并穿行解析树的解析器。结果得到的解析器发出各种类型、函数定义/调用以及与程序的结构有关的其它数据。用ANTLR4生成解析器提取的底层属性包括复杂类型/结构、循环不变量/计数器(例如,来自for each范例)以及结构化注释(例如,形式前置/后置条件语句)。示例性实施方式可以将此提取数据映射到LLVM IR中的其引用位置,因为文件名、行和列号信息存在于解析器和LLVM IR两者之中。

[0076] 本发明的示例性实施方式可以通过从源软件文件提取一串字符(诸如内联注释)来自动地确定一个或多个元数据产物。其它实施方式从文件系统或源控制系统中自动地确定元数据产物。

[0077] 分级产物间关系

[0078] 图3是图示出根据本发明的实施方式的用于软件文件的产物之间的分级关系的框图。示例性实施方式可以保持并利用这些分级产物间关系。此外,不同的实施方式可以使用不同的模式和不同的分级关系。针对图3的示例性实施方式,产物分级结构的顶部是LTS产物310。每个LTS节点310可以映射到函数和特定可变状态的集合或子集。在LTS产物310下面的是CG产物320。每个CG节点320可以映射到具有CFG产物330的特定函数,其边可以包含循环不变量和分支语义330。每个CF节点330可以包含基本块以及DT 340。在那些产物下面的是变量、常量、UD/DU链以及IR指令350。图3清楚地图示出产物可以从描述动态信息的范围的LTS节点向下直至单独的IR指令,而被映射到分级结构的不同层级。这些分级关系可以被示例性实施方式用于多种用途,包括更高效地搜索匹配产物,诸如通过首先比较更接近于分级结构顶部的产物(与更接近于底部的产物相比),从而根据高层级产物是否是匹配产物而包括或排除与高层级产物相关联的低层级产物的整体集合。附加实施方式还可以在针对缺陷或针对特征增强而定位或建议修复代码(包括通过在分级结构中上升以定位针对具有匹配的高层级产物的缺陷的修复代码)时使用分级关系。

[0079] 图4是图示出用于提供用于软件文件产物的语料库的系统的示例性实施方式的框图。示例性实施方式可以具有能够与具有多个软件文件的源430通信的接口420。针对特定

实施方式，此接口420可以被通信耦合到本地源430，诸如本地硬盘驱动器或磁盘。在其它实施方式中，接口420可以是用于通过公共或私有网络来获得文件的网络接口420。这些软件文件的公共源430的示例包括GitHUB、SourceForge、BitBucket、GoogleCode或Common Vulnerabilities and Exposures(公共漏洞和暴露)系统。私有源的示例包括公司的内部网络和存储在其上面的文件，包括在共享网络驱动器和私有储存库中的那些。本示例性系统还具有一个或多个处理器410，其被耦合到接口420以从源430获得所述多个软件文件。处理器410还可以用来针对所述多个软件文件中的每一个确定所述多个产物。这些产物可以是静态产物、动态产物、导出产物和/或元数据产物。针对附加实施方式，处理器410还可以被配置成将软件文件的每一个转换成中间表示并根据该中间表示来确定产物。

[0080] 示例性系统还具有一个或多个存储设备440a-440n，其用于存储用于软件文件的每一个的产物，并且被耦合到处理器410。这些存储设备440a-440n可以是硬盘驱动器、硬盘驱动器阵列、其它类型的存储设备以及分布式储存器，诸如通过采用Hadoop文件系统(HDFS)上的Titan和Cassandra所提供的存储器。同样地，示例性系统可以具有一个处理器410，或者采用分布处理并具有超过一个处理器410。其它实施方式还提供在接口420与存储设备440a-440n之间的直接通信耦合。

[0081] 图5是图示出用于定位设计模式的方法的示例性实施方式的框图。设计模式的示例包括错误、修复、漏洞、安全补丁、协议、协议扩展、特征以及特征增强。每个设计模式可以与在软件项目分级结构的各种层级处提取的产物(例如，规范、CG、CFG、Def-Use链、指令序列、类型以及常量)相关联。

[0082] 示例性方法提供对具有与多个软件文件510相对应的多个产物的数据库的访问。该数据库可以是图形数据库、关系数据库或平面文件。数据库可以位于本地、在私有网络上或者经由因特网或云是可访问的。一旦数据库已被访问，本方法然后可以基于用于所述多个文件520中的第一文件的所述多个产物中的至少一个而自动地识别设计模式。针对一些示例性实施方式，所述多个产物中的每一个可以是静态产物、动态产物、导出产物或元数据产物。其它实施方式可以具有不同类型的产物的混合体。此外，文件的格式不受限制，并且可以是例如二进制代码格式、源代码格式或中间表示(IR)格式。

[0083] 针对一些实施方式，可以通过进行开发产物的关键字搜索或自然语言搜索来识别设计模式。例如，源代码文件的修订版中的内联代码注释可以是识别被发现并改正的缺陷。注释可以使用诸如缺陷、错误、差错、问题、瑕疵或小故障之类的单词。这些单词可以在对元数据的关键字搜索中使用。提交日志还可以包括描述新修订版和补丁为何被应用从而解决缺陷或增强特征的文本。此外，可以将训练和反馈应用于搜索以细化搜索努力。

[0084] 附加示例性实施方式可以从CVE源搜索开发产物，其识别文本中的公共漏洞和差错并可以描述缺陷和可用修复，如果有的话。此文本可以作为产物而获得并存储在数据库中。一些源还将缺陷编码，使得代码可以被用作关键字来定位哪个文件包含缺陷。另外，可以在识别软件文件时考虑产物的源并进行加权。例如，在没有出处或内联注释的情况下，CVE源可以在识别缺陷时比储存库更加可靠。其它实施方式可以使用诸如文件名和修订号之类的元数据产物来至少初步地识别软件文件，并且基于匹配附加产物(诸如CG或CGF)来确认该识别。

[0085] 本发明的特定实施方式执行示例性方法并尝试识别用于一些、大多数或所有源代

码和LLVM IR文件的设计模式。另外，每当文件被添加到语料库时，一些实施方式访问数据库并尝试识别任何设计模式。一些实施方式还可以标记已识别设计模式以供后来使用。

[0086] 一些实施方式还找到与也已被存储在数据库中的文件相关联的源代码或LLVM IR中的缺陷的位置。例如，开发产物可以指定在源代码中的什么位置存在缺陷和在补丁中的什么位置存在修复。并且，可以分析源代码或LLVM IR并将其与具有缺陷和文件新修复版本的文件相比较，以便将差异隔离并辨别缺陷和修复位于哪里。针对特定实施方式，还可以使用在开发产物中识别的缺陷的类型来缩窄针对缺陷位置的代码搜索。附加实施方式还可以诸如使用标签来识别设计模式，并且将标识符存储在用于文件的数据库中。这允许更容易地针对一些缺陷或缺陷类型搜索数据库。此类标签的示例包括从用于软件文件的开发产物或者从源代码获得的字符串。这种方法可以应用于识别特征和特征增强并将其标记。

[0087] 针对特定示例性实施方式，设计模式位于软件文件中。针对特定示例性实施方式，设计模式可以涉及文件之间的交互，诸如接口。示例性实施方式可以通过使识别基于用于多个软件文件(诸如第一和第二文件，两者都属于一软件项目)的产物，来自动地识别设计模式。例如，可以将表示设计模式的预先识别模式(诸如接口失配误差)存储在数据库中，或者存储在允许将来自第一和第二文件的产物用来识别针对这些文件存在接口误差的其他位置。针对示例性实施方式的示例性设计模式包括缺陷、修复、特征、特征争抢或预先识别的程序片段。

[0088] 针对特定示例性实施方式，本方法在产物中定位表示缺陷或修复的字符串。常常地，在开发产物中存在此类串(诸如错误、差错或缺陷)，以及关于修复和在代码中什么位置可以找到那些修复的串。这些开发产物还可以具有表示特征或特征增强的串。

[0089] 针对一些示例性实施方式，设计模式基于表示设计模式的预先识别模式。这些预先识别模式可以由用户创建，可以是先前由与本公开相关联的方法识别，或者可以被以某种其它方式识别。这些预先识别模式可以对应于缺陷、修复、特征、特征增强或者感兴趣或具有其它重要性的项。

[0090] 图6是图示出用于定位缺陷的方法的示例性实施方式的流程图。本方法包括访问具有对应于多个软件文件的多个软件产物的数据库610，诸如语料库。然后，对产物进行分析以从大量数据中辨别模式。例如，此分析可以包括将所述多个产物620进行聚类。通过将数据进行聚类，可以找到未被已知包含已知缺陷的文件中的已知缺陷。因此，根据该聚类，示例性方法可以基于一个或多个先前识别的缺陷630来识别先前未识别的缺陷。

[0091] 本发明的一些示例性实施方式可以对语料库采用机器学习。机器学习涉及通过从低层级产物开始学习数据的分级结构以捕捉数据中的相关特征，并且然后构建更复杂的表示。一些示例性实施方式可以对语料库采用深度学习。深度学习是基于学习数据表示的机器学习方法的较宽泛家族的子集。针对一些实施方式，可以将自动编码器用于聚类。

[0092] 针对特定示例性实施方式，可以通过一组自动编码器来处理产物，以自动地发现未标记图形和文档产物的紧凑表示。图形产物包括可以用图形式表示的那些产物(诸如CG、CFG、UD链、DU链以及DT)。然后可以将图形产物的紧凑表示聚类，以发现软件设计模式。从相应元数据产物提取的知识可以用来标记设计模式(例如，错误、改正、漏洞、安全补丁、协议、协议扩展、特征以及特征增强)。

[0093] 针对特定示例性实施方式，自动编码器是结构化稀疏自动编码器(SSAE)，其可以

将矢量取作输入并提取公共特征。针对用以自动地发现程序的特征的一些实施方式，首先以矩阵形式表示提取的图形产物。可以将提取产物中的许多表示为邻接矩阵，包括例如CFG、UD链以及DU链。可以在软件文件和项目分级结构的每个层级处学习结构特征。

[0094] 在图形产物中的节点的数目可以大范围地改变；因此，可以将中间产物提供为用于深度学习的输入。一个此类中间产物是图的拉普拉斯的前k个特征值，以使得能够实现深度学习以与谱聚类类似地执行处理。其它中间产物包括聚类系数，其提供图中的节点趋向于聚集在一起的程度的度量，诸如全局聚类系数、网络平均聚类系数以及传递性比。另一中间产物是图的荫度，即图有多稠密的度量。具有许多边的图具有高荫度，并且具有高荫度的图具有稠密子图。另一中间产物是等周数，即图是否具有瓶颈的数值度量。这些中间产物捕捉图结构的不同方面以便在机器学习方法中使用。

[0095] 针对示例性实施方式，机器学习(包括深度学习)可以采用使用从简单的自动编码器结构开始的多步过程并迭代地细化本方法以开发SSAE来训练的算法。还可以训练SSAE以从中间产物学习特征。自动编码器学习未标记数据的紧凑表示。可以用学习至恒等函数的近似的神经网络(包括至少一个隐藏层且具有相同数目的输入和输出)来对其建模。自动编码器将输入信号脱水(dehydrate)(编码)成一组基本的描述参数，并将那些信号再水合((rehydrate))(解码)以重新创建原始信号。可以在训练期间自动地选择该描述参数以优化对所有训练信号的再水合。脱水信号的基本性质提供了用于将信号分组成聚类的基础。

[0096] 自动编码器可以通过将输入信号映射到较低维数特征空间来减小输入信号的维数。示例性实施方式然后可以在由自动编码器发现的特征空间中执行代码的聚类和分类。k均值算法对学习的特征进行聚类。k均值算法是一种迭代细化技术，其将特征划分成k个聚类，这使结果得到的聚类均值最小化。可以基于所提取的主题的数目来选择聚类的初始数目k。在该数目的潜在聚类内进行搜索从而针对许多不同k中的每一个计算新结果是非常高效的，因为用于k均值聚类的操作度量是基于欧几里德距离。示例性实施方式可以使用在从其导出聚类特征的软件文件内最频繁地出现的主题的标签将结果得到的聚类进行分类。

[0097] 虽然特征向量是稀疏且紧凑的，但是可能难以仅仅通过特征向量的检查来理解输入向量。因此，示例性实施方式可以利用与先前学习的权值参数相关联的先验知识。给定充分的语料库，例如针对“已修复”代码，参数空间中的模式应当出现。示例性实施方式可以使用由被收集到该点的数据集给定的先验信息来将特定模式结合到自动编码器中。特别地，当标签被系统获悉时，示例性实施方式可以将该信息结合到自动编码器操作中。

[0098] 示例性实施方式可以使用数据库管理(例如，联合，滤波)和分析操作(例如，奇异值分解(SVD)、双聚类)的混合。示例性实施方式的图理论(例如，谱聚类)和机器学习或深度学习算法两者都可以将类似的算法原语用于特征提取。还可以使用SVD来对用于学习算法的输入进行降噪，并使用较少的维度来近似数据，并且因此执行数据简化。

[0099] 示例性实施方式可以通过文档产物的无监督语义标签生成(包括经由文本分析)，而封装随时间推移且跨各程序的代码状态的人类理解。文本分析的示例是隐含狄利克雷分配(LDA)。可以使用LDA和主题建模来从文档产物提取语义信息。这些方法是着眼于单词或短语的出现、忽视顺序的“词袋(bag-of-words)”技术。例如，表示“科学计算”的袋子可以具有诸如“FFT”、“小波”、“sin”、“atan”之类的种子术语。示例性实施方式可以使用从源提取的文档产物，诸如源注释、CG/CGF节点标签和提交消息，以通过对术语的出现进行计数来填

充“袋子”。结果得到的固定bin直方图可以被馈送到受限玻尔兹曼机(RBM)，即适合于文本应用的深度学习算法的实现方式。提取的主题捕捉与提取的文档产物相关联的语义信息，并且可以充当用于经由自动编码器由图产物的无监督学习而形成的聚类的标签(例如，错误/改正、漏洞/补丁)。可以由附加示例性实施方式采用的其它形式的文本分析包括自然语言处理、词法分析以及预测分析。

[0100] 从文档产物提取的主题标签可以提供标记信息以告知自动编码器的结构化。示例性实施方式可以基于学习主题、表示顺序软件模式(即，在软件修订版之前/之后)的语义公共性，针对训练数据群体查询语料库数据库。这些模式可以捕捉嵌入软件开发文件(诸如在提交日志、改变日志和注释)中的变化，其随时间推移而与软件开发生命周期相关联。这些变化的关联提供对与检测和修复(诸如错误/改正、漏洞/安全补丁以及特征/增强)相关的软件的演进的洞悉。此信息还可以用来理解并标记自动地从产物语料库提取的知识。

[0101] 图7示出了根据本发明的实施方式的用于识别设计模式的产物的聚类的框图。可以在软件文件分级结构的每个层级(包括系统、程序、函数以及块710)处学习结构特征。可以针对聚类715分析图形产物，诸如CG、CFG以及DT。可以将这些图形产物变换成图形不变量特征720。这些图形特征740然后可以作为输入提供给图形分析模块760，诸如自动编码器，并且针对类似的设计模式(其被聚集在一起780)检查结果得到的聚类。可以将文本(诸如来自源代码或来自开发产物的一个或多个字符串)映射到标签730。可以由文本分析模块770诸如通过使用LDA或其它自然语言处理来分析这些标签760，并且可以将标签与从其导出该标签的相应发现聚类780相关联。可以用软件、硬件或其组合来实现这些模块760、770。

[0102] 图8示出了用于使用语料库来识别软件文件的方法的示例性实施方式的流程图。该示例性实施方式获得软件文件810。该文件可以经由网络接口从诸如经由因特网、云或私人公司的服务器的公共储存库的公共或私有源获得。一些示例性实施方式还可以从诸如本地硬盘驱动器、便携式硬盘驱动器或磁盘的本地源获得软件文件。示例性实施方式可以从源获得单个文件或多个文件，并可以诸如经由脚本语言的使用而自动地或通过用户交互而手动地这样做。本示例性方法然后可以针对软件文件820确定多个产物，诸如本文所述的任何其它产物。本示例性方法然后可以访问数据库830，其存储用于多个参照软件文件中的每一个的多个参照产物。该参照产物可以被存储在语料库数据库中。针对特定示例性实施方式，这些参照文件可以包括先前已经获得且针对一些实施方式其产物已连同软件文件一起被存储在数据库中的软件文件。将针对所获得的软件文件确定的产物或其多个子集与存储在数据库850中的参照产物或其多个子集相比较。示例性实施方式可以通过识别具有与所述多个产物850匹配的所述多个参照产物的参照软件文件来识别软件文件。由于比较的产物和参照产物匹配，所以软件文件和参照软件文件被识别为同一文件。

[0103] 然后，还可以比较附加产物或代码部分，以增加进行了正确识别的置信度水平。该置信度可以是固定的或可调整的，并且可以基于多种标准，诸如匹配的产物的数目、哪些产物匹配以及数目和哪些产物的组合。例如，可以针对特定的数据集及其观察进行此调整。此外，针对一些实施方式，匹配可以包括模糊匹配(诸如针对小于100％匹配的百分比具有可调整设置)以使匹配被声明。

[0104] 针对特定示例性实施方式，可以在匹配和识别过程中对一些产物给定更多或更少的权值。例如，可以对公共产物(诸如指令是与32位还是64位处理器匹配)给定零的权值或

某个其它较小权值。一些产物在变换下可以或多或少地是不变的，并且针对特定实施方式可以相应地调整用于这些产物的权值。例如，可以认为文件名或CG产物在确立文件的标识时是非常富含信息的，而特定产物(诸如LTS或DT)针对特定示例性实施方式和源可能被认为不那么具有决定性，并被给定较小的权值。附加实施方式可以对产物的一些组合给定更大的权值，以在进行比较时识别匹配。例如，具有CFG和CG产物匹配可以在进行识别时被给定比具有基本块产物和DT产物匹配更多的权值。同样地，可以在文件的识别时，对不匹配的一些产物给定更多或更少的权值。识别过程中评估加权的附加示例可以包括诸如使用匹配产物的百分比或某个其它度量，来表示识别阈值。附加阈值可以改变识别阈值，包括基于诸如文件的源、文件的类型、时间戳(其哪些包括文件的日期)、文件的尺寸或一些产物是否对于文件而言不能确定或者以其他方式不可用之类的事项。

[0105] 附加实施方式可以通过将软件文件转换成中间表示(诸如LLVM IR)并根据该中间表示来确定所述多个产物中的至少一个来针对软件文件确定所述多个产物中的一些。其它实施方式可以通过从软件文件(诸如源代码文件或文件编制文件)提取字符串，来确定所述多个产物中的一些。

[0106] 示例性实施方式还可以包括通过分析与已识别参照软件文件相关联的参照产物中的至少一个，来确定是否存在软件文件的新版本。例如，一旦已经识别了软件文件，则可以检查数据库以查看软件文件的新修订版是否可用，诸如通过检查相应参照文件的修订号或时间戳，或者可以将参照文件识别为另一文件的较旧修订版的与数据库中的产物和文件相关联的标签。附加示例性实施方式还可以自动地提供软件文件的新版本，包括向用户或者公共或私有源提供。

[0107] 特定附加实施方式可以通过分析与已识别参照软件文件相关联的参照产物中的至少一个来确定是否存在软件文件的补丁。例如，示例性实施方式可以检查与参照软件文件相关联的产物，并确定针对该文件存在补丁，包括尚未被应用于软件文件的补丁。附加实施方式可以自动地对软件文件应用补丁或者提示用户关于他们是否想要应用补丁。

[0108] 特定附加实施方式可以分析该补丁，且针对一些实施方式还分析软件文件(或参照软件文件，因为其是匹配的)，以确定对应于软件文件中的缺陷的修复的补丁的修复部分。针对一些实施方式，此分析可以在获得软件文件之前或之后发生。附加实施方式可以仅将补丁的修复部分应用于软件文件，包括自动地或者提示用户关于他们是否想要应用补丁的修复部分。附加实施方式可以将补丁的该修复部分提供给源以使其在源处被应用。此外，补丁和软件文件的分析可以包括将补丁和软件文件转换成中间表示，并根据该中间表示来确定所述多个产物中的至少一个。同样地，附加实施方式可以分析补丁和软件文件(或参照软件文件，因为其是匹配的)，以确定与软件文件中的特征的改进或改变相对应的补丁的特征增强部分。附加实施方式可以仅将补丁的特征增强部分应用于软件文件，包括自动地或者提示用户他们是否想要应用补丁的特征增强部分。

[0109] 附加实施方式可以通过分析与已识别参照软件文件相关联的参照产物中的至少一个，来确定在软件文件中是否存在缺陷。例如，参照软件文件可以具有将其识别为具有缺陷(对于其而言有修复可用)的产物。附加实施方式可以自动地修复软件文件中的缺陷，包括通过自动地用源代码的修复块来替换源代码块，或者用中间表示的修复修复块来替换软件文件中的中间表示块。附加实施方式可以通过用二进制补丁替换二进制的一部分来修复

二进制文件中的缺陷。针对特定实施方式，可以将已修复文件发送到软件文件的源。附加实施方式可以允许向软件文件的源提供修复代码，以便在那里修复文件。

[0110] 图9是图示出用于识别代码的方法的示例性实施方式的流程图。示例性方法可以获得一个或多个软件文件910。针对软件文件，可以确定多个产物920。如果产物已被确定的话，则特定实施方式可以替代地获得产物而不是确定产物。可以访问存储多个参照产物的数据库930。参照产物是如本文所述的产物，并且可以对应于参照软件文件、参照设计模式或其它感兴趣代码块。数据库可以被存储在许多位置上，诸如在本地或者在网络驱动器上，或者通过因特网或在云中可访问，并且还可以跨多个存储设备分布。然后，可以通过将对应于程序片段的所述多个产物与对应于程序片段的所述多个参照产物匹配，来识别在一个或多个软件文件中或者与他们(诸如接口错误)相关联的程序片段940。程序片段是文件、程序、基本块、函数或函数之间的接口的子部分。程序片段可以小到单个指令或者大到整个文件、程序、基本块、函数或接口。所选部分可以足以以任何期望的置信度来识别程序片段，所述期望置信度对于一些实施方式而言可以是设定的或可调整的，并且其可以改变，诸如上文相对于识别文件所述。

[0111] 针对附加实施方式，针对软件文件确定产物包括将软件文件的每一个转换成中间表示并根据该中间表示来确定产物中的至少一个。针对一些实施方式，软件文件和参照软件文件每个采取源代码格式，或者每个采取二进制代码格式。针对附加实施方式，程序片段对应于软件文件中的缺陷，并且已在数据库中被识别为对应于缺陷。附加实施方式可以自动地修复软件文件中的缺陷或者向用户提供一个或多个修复选项以修复该缺陷。特定实施方式可以将修复选项排序，包括例如基于由用户选择的一个或多个先前修复选项或者基于用于修复选项的成功的可能性进行排序。

[0112] 图10是图示出根据本发明的实施方式的使用软件文件的数据库语料库的系统的框图。示例性系统包括可以与具有至少一个软件文件的源1010进行通信的接口1020。接口1020还被通信耦合到处理器1030。针对附加实施方式，接口1020还可以被直接地耦合到存储设备1040。存储设备1040可以是多种众所周知的存储设备或系统，诸如网络或本地存储设备，诸如例如单个硬盘驱动器或者具有多个硬盘驱动器的分布式存储系统。存储设备1040可以存储参照产物，包括针对许多参照软件文件中的每一个，并且可以被通信耦合到处理器1030。处理器1030可以被配置成促使从源1010获得软件文件。此软件文件的身份和是否存在可用的软件的新版本、是否存在可用的补丁或该文件是否包含缺陷或未增强特征都是示例性系统可以解决的问题的示例。处理器1030还被配置成针对软件文件确定多个产物，访问存储设备1040中的参照产物，将用于软件文件的产物与存储在存储设备1040中的参照产物相比较，并且通过识别具有与用于软件文件的已比较产物相对应的参照产物的参照软件文件来识别软件文件。

[0113] 在示例性系统的附加实施方式中，处理器1030可以被配置成如果对于该文件而言一个补丁在存储设备1040中可用的话，则自动地对软件文件应用补丁。在附加实施方式中，处理器还可以被配置成分析已识别补丁和软件文件，以确定在软件文件中是否存在与缺陷的修复相对应的补丁的修复部分，并且如果是这样的话，自动地仅将补丁的该修复部分应用于软件文件，或者提示用户。

[0114] 图10的框图还可以图示出根据本发明的实施方式的使用数据库语料库的另一示

例性系统。另外示出的本示例性系统包括可以与具有一个或多个软件文件的源1010通信的接口1020。接口1020还被通信耦合到处理器1030。针对附加实施方式，接口1020还可以被直接地耦合到存储设备1040。存储设备1040可以是多种众所周知的存储设备或系统，诸如联网或本地存储设备，诸如单个硬盘驱动器或者具有多个硬盘驱动器的分布式存储系统。存储设备1040可以存储参照产物，并且可以被通信耦合到处理器1030。处理器1030可以被配置成促使获得一个或多个软件文件，以针对一个或多个软件文件确定多个产物，访问存储多个参照产物的数据库，并且通过将对应于程序片段的所述多个产物与对应于程序片段的所述多个参照产物相匹配，来识别用于一个或多个软件文件的程序片段。针对特定示例性实施方式，程序片段已在数据库中被识别为对应于缺陷。此类缺陷的示例包括错误、安全漏洞以及协议瑕疵。这些缺陷可以在一个或多个软件文件内，或者可以与软件文件之间的一个或多个接口相关。附加实施方式还可以使处理器被配置成自动地修复一个或多个软件文件中的缺陷。针对特定示例性实施方式，程序片段已在数据库中被识别成对应于特征，并且特定实施方式还可以自动地提供特征增强，包括以用于源代码或二进制文件的补丁的形式。

[0115] 修复

[0116] 示例性实施方式支持用于自动化修复的程序合成，包括通过替换CG节点(函数)、CFG节点(基本块)、特定指令或特定变量和常量，来将所选修复实例化。这些元素(例如，函数、基本块、指令)可与具有兼容接口的元素(即，相同数目的参数、类型和输出)相交换，并且可以通过用LLVM IR的修复块替换LLVM IR的缺陷块来变换LLVM IR。

[0117] 特定实施方式还可以选择将基本块与函数调用交换以及将函数调用与一个或多个基本块交换。特定实施方式可以修补源代码和二进制码。附加实施方式还可以在用于交换的适当元素不存在时创建该元素。可以使用高层级产物(例如，LTS和Z谓词)来导出用于软件补丁的兼容实现方式。示例性实施方式可以利用提取的图形表示的分级结构，首先将分级结构升至修复模式的适当表示，并且然后从分级结构降至(经由编译)具体实现方式。产物的分级性质可以帮助构成修复代码。

[0118] 示例性实施方式可以允许用户限制目标程序(源或二进制)，并且示例性实施方式发现任何缺陷设计模式的存在。针对每个缺陷，可以向用户提供修复策略(即，修复设计模式)。用户可以选择用于将修复合成并修补目标的策略。特定示例性实施方式还可以学习用户选择以将未来的修复解决方案最佳地排序，并且还可以按照排列顺序向用户呈现修复策略。特定实施方式还可以自主地运行，修复整个软件语料库内的缺陷或漏洞，包括连续地、周期性地和/或在设计环境中进行修复。

[0119] 除上文所讨论的实施方式之外，可以针对多种用途利用本发明。例如，可以在软件代码的编程期间使用示例性实施方式以辅助程序员，包括识别缺陷或建议代码重用。可以将附加示例性实施方式用于发现缺陷和漏洞并可选地自动地将其修复。可以使用其它示例性实施方式来优化代码，包括识别未使用的代码、低效的代码以及用以替换不那么高效的代码的推荐代码。

[0120] 示例性实施方式还可以被用于风险管理和评估，包括相对于在一些代码中可能存在什么漏洞。还可以在设计认证过程中使用附加实施方式，包括提供软件文件没有已知缺陷(诸如错误、安全漏洞以及协议缺陷)的证明。

[0121] 本发明的其它附加示例性实施方式包括:代码重用发现器(在代码库中找到执行同样事情的代码)、代码质量测量、文本描述至代码转换器、库生成器、测试案例生成器、代码-数据分离器、代码映射和探索工具、现有代码的自动架构生成、架构改进建议器、错误/差错估计器、无用代码发现、代码-特征映射、自动化补丁审阅器、代码改进决策工具(将特征列表映射到最小变化)、对现有设计工具(例如,企业设计师)的扩展、替换实现建议器、代码探索和学习工具(例如,用于教学)、系统级代码许可足迹以及企业软件使用映射。

[0122] 应当理解的是,可以用许多不同方式来实现上文所述示例性实施方式。在一些情况下,本文中所述的各种方法和机器每个可以用具有中央处理机、存储器、磁盘或其它大容量储存器、(多个)通信接口、(多个)输入/输出(I/O)设备及其它外围设备的物理、虚拟或混合式通用计算机实现。该通用计算机被变换成执行上述方法的机器,其例如通过将软件指令加载到数据处理器中且然后引起指令的执行以执行本文中所述功能。还可以将软件指令模块化,诸如具有用于摄取文件以形成语料库的摄取模块、用以确定用于针对语料库的文件和/或要针对设计模式而被识别或分析的文件的产物的分析模块、用以执行机器学习的图形分析模块和文本分析模块、用于识别文件或设计模式的识别模块以及用于修复代码或提供已更新或已修复文件的修复模块。针对特定示例性实施方式,可以将这些模块组合或分离成附加模块。

[0123] 如在本领域中已知的,此类计算机可以包含系统总线,其中,总线是被用于计算机或处理系统的组件之间的数据传输的一组硬件线。总线或多条总线本质上是共享导管,其连接计算机系统的不同元件,诸如处理器、磁盘储存器、存储器、输入/输出端口、网络端口等,这使得能够实现元件之间的信息传输。一个或多个中央处理器单元被附接到系统总线并提供计算机指令的执行。还被附接到系统总线的通常是用于将各种输入和输出设备(例如,键盘、鼠标、显示器、打印机、扬声器等)连接到计算机的I/O设备接口。网络接口允许计算机连接到被附接到网络的各种其它设备。存储器提供用于被用来实现实施方式的计算机软件指令和数据的易失性存储。磁盘或其它大容量储存器提供用于被用来实现例如本文中所述的各种程序的计算机软件指令和数据的非易失性存储。

[0124] 实施方式因此通常可以用硬件、固件、软件或其任何组合来实现。此外,示例性实施方式可以完全或部分地驻留于云上,并且可以经由因特网或其它联网架构而可访问。

[0125] 在一些实施方式中,本文中所述的进程、设备以及过程组成计算机程序产品,包括非瞬态计算机可读介质,例如可移动存储介质,诸如一个或多个DVD-ROM、CD-ROM、磁盘、磁带等,其提供用于系统的软件指令的至少一部分。可以用任何适当的软件安装程序来安装此类计算机程序产品,如在本领域中众所周知的。在另一实施方式中,还可以通过线缆、通信和/或无线连接来下载软件指令的至少一部分。

[0126] 此外,在本文中可以将固件、软件、例程或指令描述为执行数据处理器的一些动作和/或功能。仍然,应认识到的是,包含在本文中的此类描述仅仅是为了方便起见,并且此类动作事实上由计算设备、处理器、控制器或执行固件、软件、例程、指令等的其它设备而引起。

[0127] 还应当理解的是,流程图、框图以及网络图可以包括被不同地布置或者不同地表示更多或更少的元件。但是进一步应理解的是,特定实施方式可以规定图示出实施方式的执行的框图和网络图及框图和网络图的数目以特定方式来实现。

[0128] 因此，还可以用多种计算机架构、物理、虚拟、云计算机和/或其某种组合来实现其它实施方式，并且因此本文中所述的数据处理器仅仅意图用于举例说明的目的，而不作为实施方式的限制。

[0129] 虽然已参照本发明的示例性实施方式特别地示出并描述了本发明，但本领域的技术人员将理解的是，在不脱离所附权利要求所涵盖的本发明的范围的情况下，可对其进行形式和细节方面的各种改变。

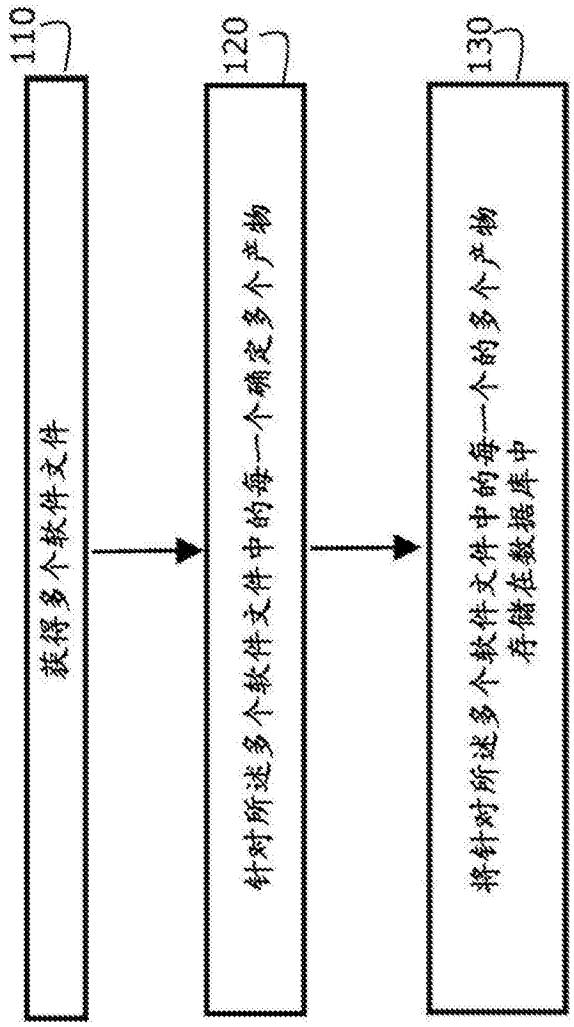

110
获得多个软件文件
120
针对所述多个软件文件中的每一个确定多个产物
130
将针对所述多个软件文件中的每一个的多个产物存储在数据库中

图1

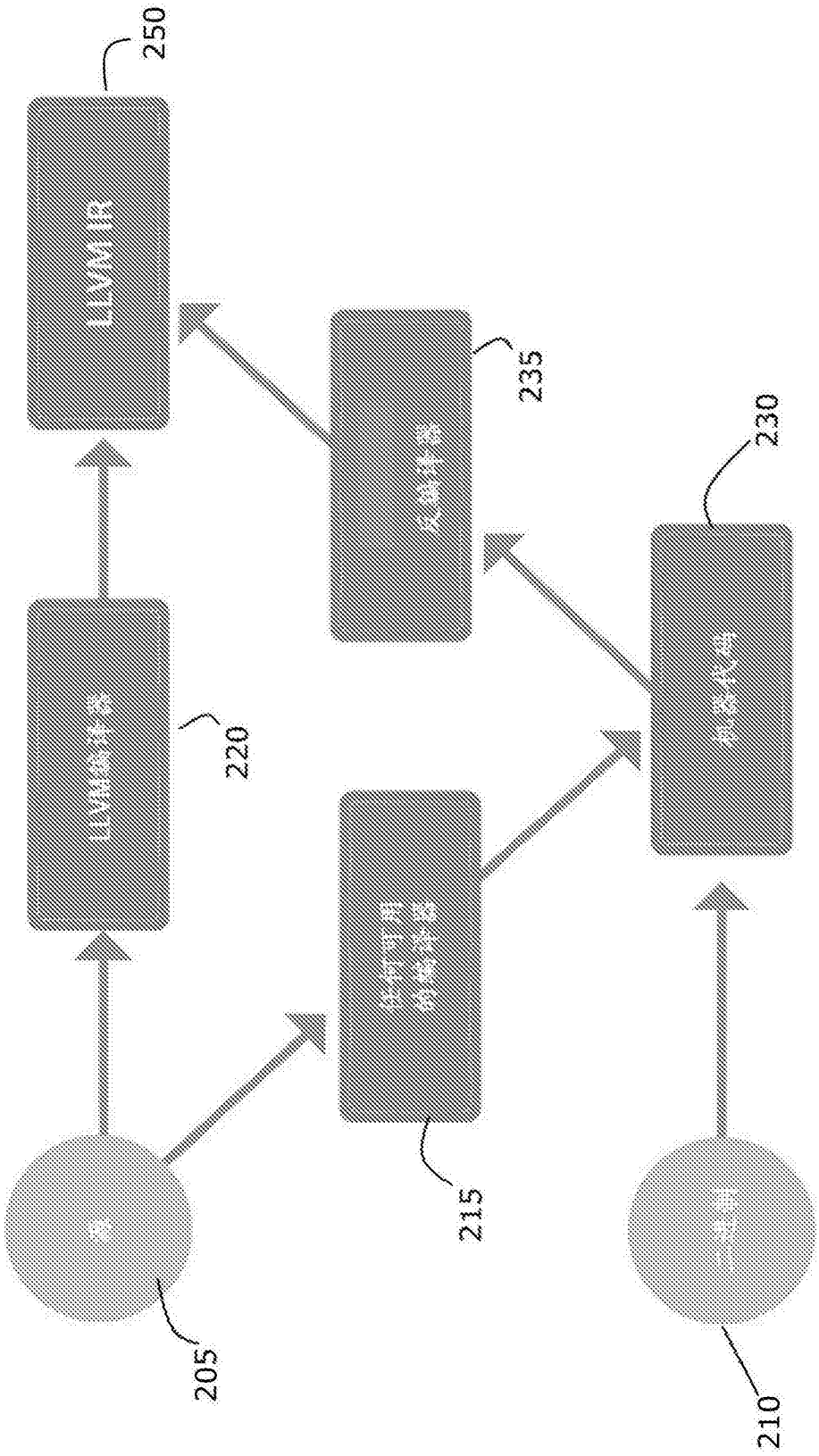


图2

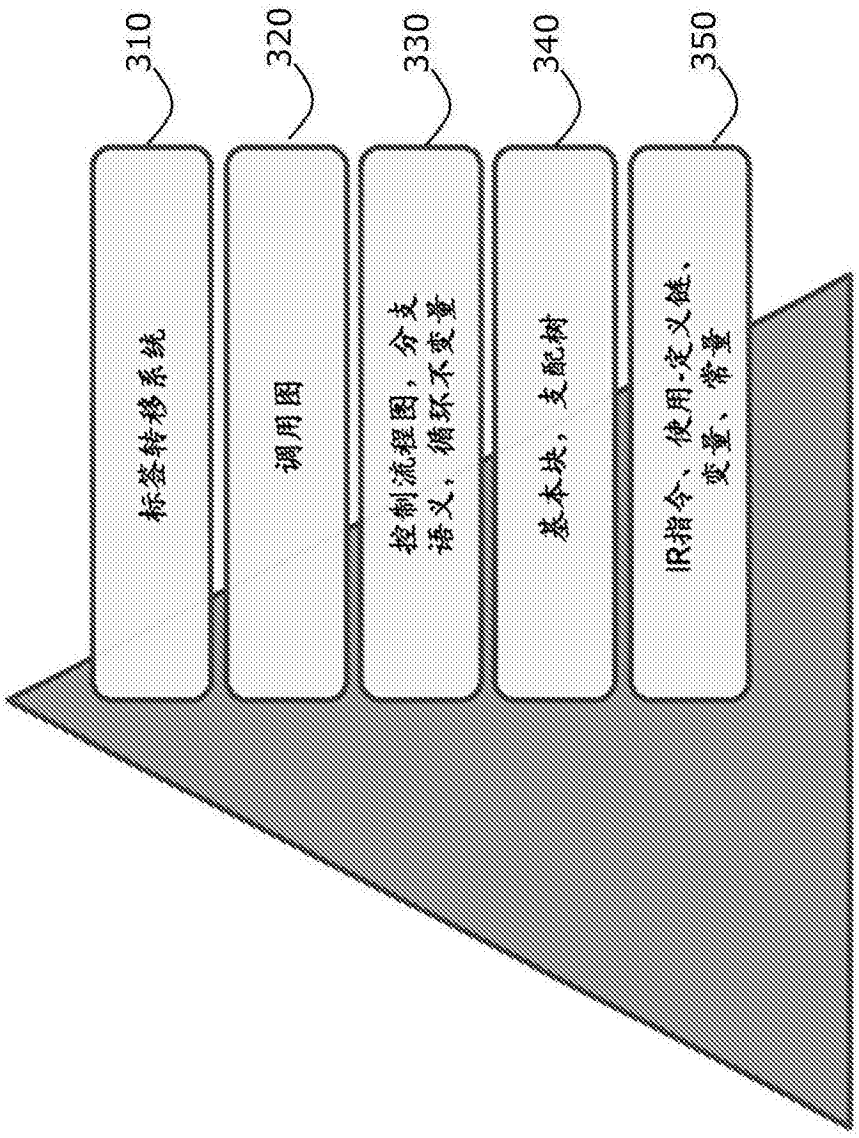

310
320
330
340
350
标签转移系统
调用图
控制流程图，分支
语义，循环不变量
基本块，支配树
IR指令、使用-定义链，
变量、常量

图3

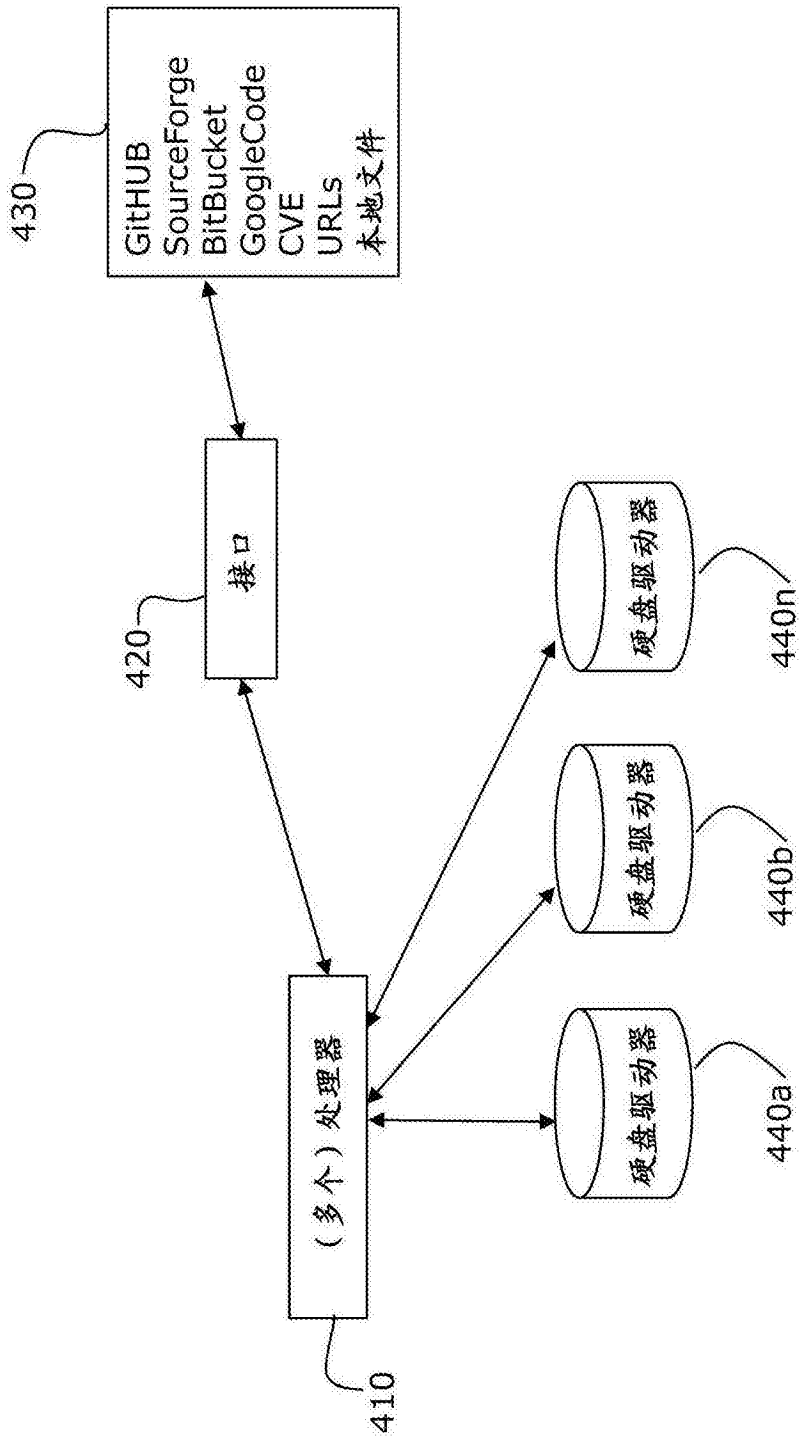

430
GitHUB
SourceForge
BitBucket
GoogleCode
CVE
URLs
本地文件
420
接口
410
（多个）处理器
硬盘驱动器
440a
硬盘驱动器
440b
硬盘驱动器
440n

图4

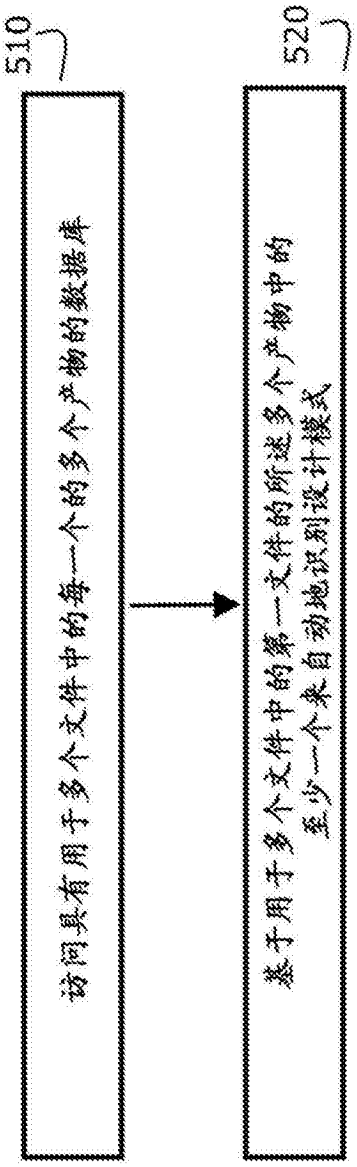

510
访问具有用于多个文件中的每一个的多个产物的数据库
520
基于用于多个文件中的第一文件的所述多个产物中的
至少一个来自动地识别设计模式

图5

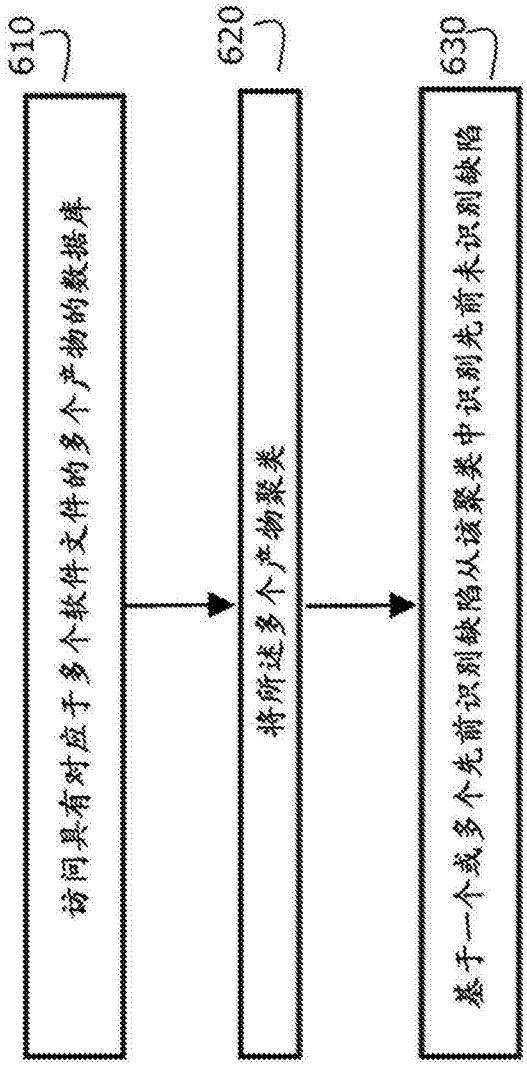

610
访问具有对应于多个软件文件的多个产物的数据库
620
将所述多个产物聚类
630
基于一个或多个先前识别缺陷从该聚类中识别先前未识别缺陷

图6

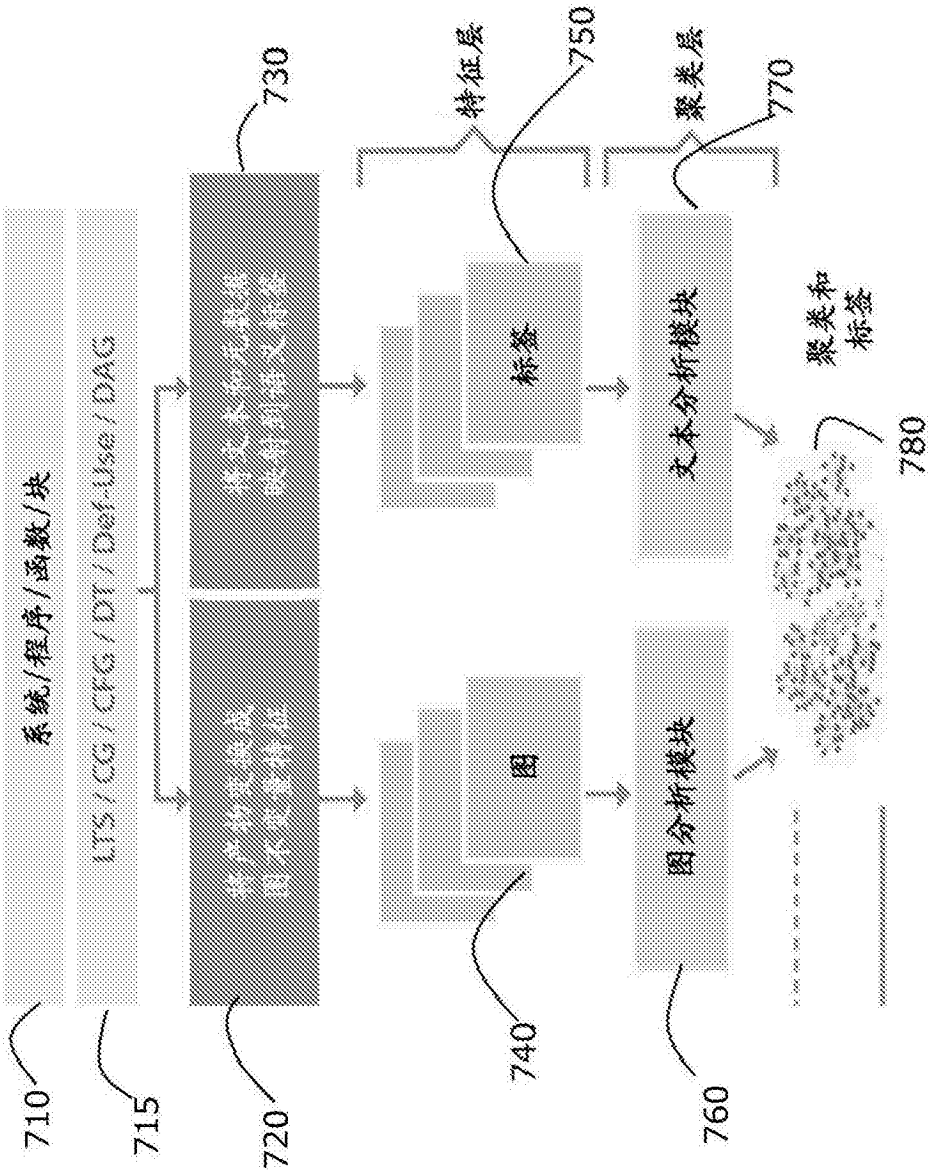

系统/程序/函数/块
LTS / CG / CFG / DT / Def-Use / DAG
710
715
720
730
740
750
760
770
780
特征层
聚类层
标签
图
文本分析模块
图分析模块
聚类和
标签

图7

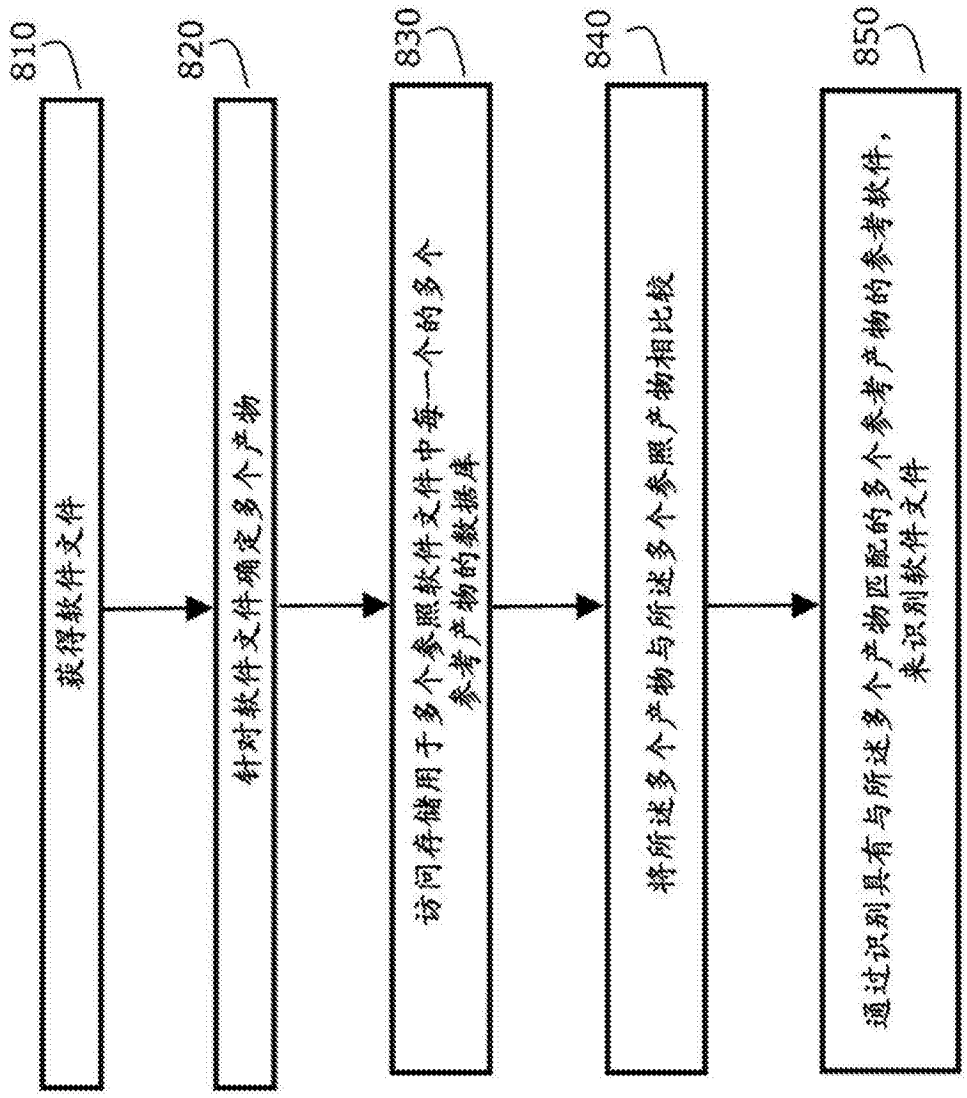

810
获得软件文件
820
针对软件文件确定多个产物
830
访问存储用于多个参照软件文件中每一个的多个参考产物的数据库
840
将所述多个产物与所述多个参照产物相比较
850
通过识别具有与所述多个产物匹配的多个参考产物的参考软件，来识别软件文件

图8

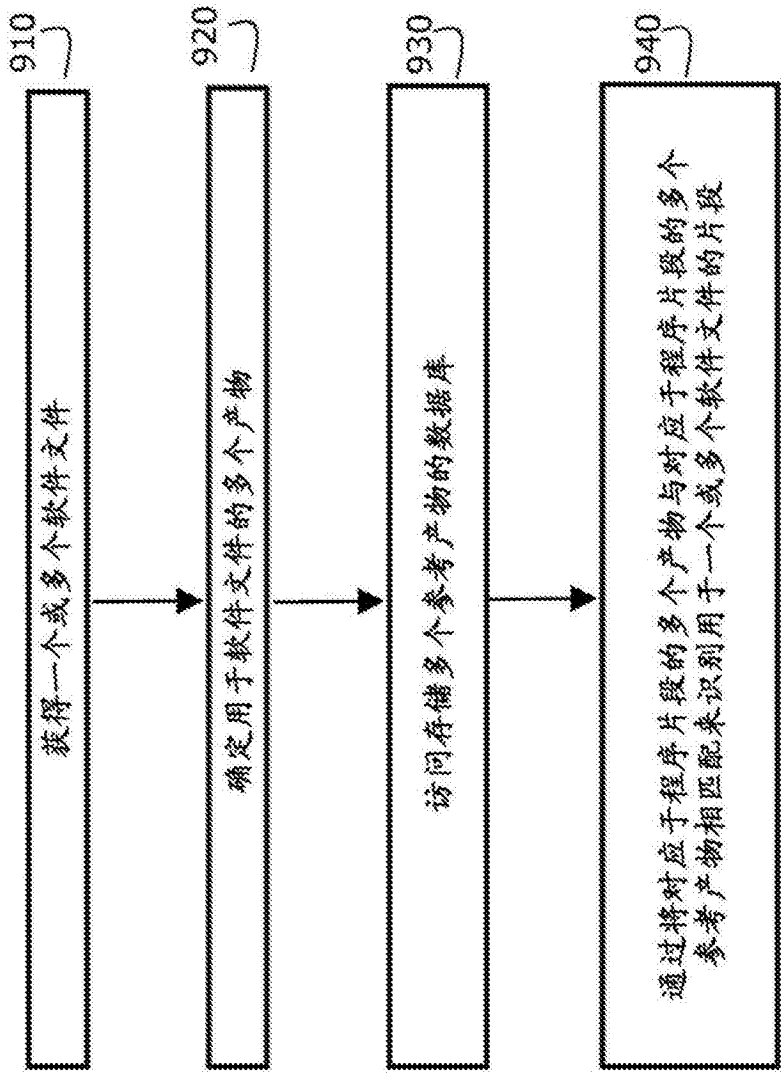

910
获得一个或多个软件文件
920
确定用于软件文件的多个产物
930
访问存储多个参考产物的数据库
940
通过将对应于程序片段的多个产物与对应于程序片段的多个参考产物相匹配来识别用于一个或多个软件文件的片段

图9

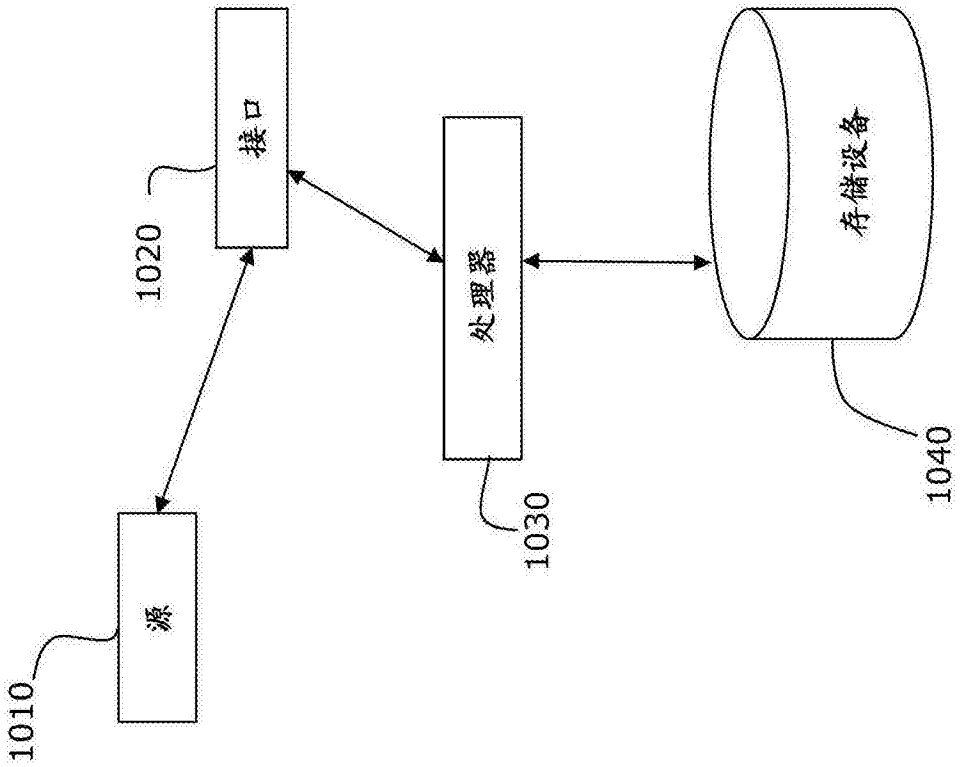


图10