(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 106537332 A
(43)申请公布日 2017.03.22

(21)申请号 201580031456.7

(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/US2015/035131 2015.06.10

(87)PCT国际申请的公布数据
WO2015/191731 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·R·什尼德曼 S·H·钱

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

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

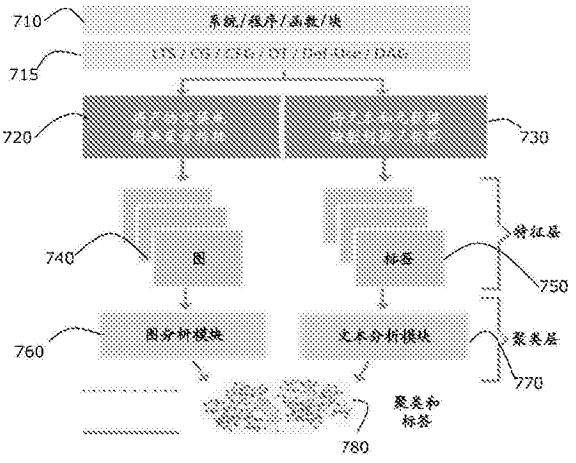
权利要求书3页 说明书16页 附图9页

(54)发明名称

软件分析系统和方法

(57)摘要

提供用于在软件中定位设计模式的系统、方法和计算机程序产品。示例方法包括：访问具有与多个软件相对应的多个产物的数据库，以及通过自动分析与软件相关联的至少一个产物来标识用于软件文件中的至少一个软件文件的设计模式。附加的实施例还提供用于将用于软件的设计模式的标识符存储在数据库中。对于某些示例实施例，产物包括开发性的，其可以搜索表示设计模式(诸如缺陷、特征或修复)的字符串。附加的示例实施例还包括在软件文件中找到实现设计模式的程序片段。



CN 106537332 A

1.一种用于标识设计模式的方法,包括:

访问具有用于多个文件中的每一个文件的多个产物的数据库;和

基于用于所述多个文件中的第一文件的所述多个产物中的至少一个产物来自动地标识设计模式。

2.根据权利要求1所述的方法,其中,所述设计模式在所述第一文件中。

3.根据权利要求1所述的方法,其中,自动地标识所述设计模式还包括:将所述设计模式的所述标识还基于用于所述多个文件中的第二文件的所述多个产物中的至少一个产物,其中,所述第一文件和所述第二文件都属于项目。

4.根据权利要求3所述的方法,其中,自动地标识所述设计模式包括:将用于所述第一文件的所述多个产物中的所述至少一个产物和用于所述第二文件的所述多个产物中的所述至少一个产物匹配到表示所述设计模式的预先标识的模式。

5.根据权利要求4所述的方法,其中,所述设计模式涉及所述第一文件和所述第二文件之间的接口。

6.根据权利要求1所述的方法,其中,所述设计模式是缺陷或修复。

7.根据权利要求1所述的方法,其中,所述设计模式是特征或特征增强。

8.根据权利要求1所述的方法,其中,所述设计模式是预先标识的程序片段。

9.根据权利要求1所述的方法,其中,基于所述多个产物中的所述至少一个产物自动地标识所述设计模式包括:在所述多个产物中的所述至少一个产物中定位表示缺陷或修复的字符串。

10.根据权利要求9所述的方法,其中,所述多个产物中的所述至少一个产物是开发产物。

11.根据权利要求1所述的方法,其中,基于所述多个产物中的所述至少一个产物自动地标识所述设计模式包括:在所述多个产物中的所述至少一个产物中定位表示特征或特征增强的字符串。

12.根据权利要求11所述的方法,其中,所述多个产物中的所述至少一个产物是开发产物。

13.根据权利要求1所述的方法,其中,基于所述多个产物中的所述至少一个产物自动地标识所述设计模式包括:将所述多个产物中的所述至少一个产物与表示所述设计模式的预先标识的模式进行匹配。

14.根据权利要求1所述的方法,其中,所述多个产物中的所述至少一个产物各自是静态产物。

15.根据权利要求1所述的方法,其中,所述多个产物中的所述至少一个产物各自是动态产物。

16.根据权利要求1所述的方法,其中,所述多个产物中的所述至少一个产物各自是导出的产物。

17.根据权利要求1所述的方法,其中,所述多个产物中的所述至少一个产物各自是元数据产物。

18.根据权利要求1所述的方法,还包括:在所述数据库中存储用于所述设计模式的标识符。

19.根据权利要求18所述的方法，其中，存储用于所述设计模式的标识符包括：使用从用于所述第一文件的所述多个产物中的至少一个产物而获得的字符串来存储用于所述设计模式的标签。

20.根据权利要求2所述的方法，还包括在所述第一文件中找到与所述设计模式相对应的程序片段。

21.根据权利要求20所述的方法，其中，所述第一文件是二进制代码格式。

22.根据权利要求20所述的方法，其中，所述第一文件是源代码格式。

23.根据权利要求20所述的方法，其中，所述第一文件是中间表示(IR)格式。

24.一种用于标识设计模式的方法，包括：

访问具有多个产物的数据库；

聚类所述多个产物；和

基于一个或多个先前标识的设计模式从所述聚类中标识先前未标识的设计模式。

25.根据权利要求24所述的方法，其中，所述先前未标识的设计模式、以及所述一个或多个先前标识的设计模式是相同的设计模式。

26.根据权利要求24所述的方法，其中，所述先前标识的设计模式是缺陷。

27.根据权利要求26所述的方法，还包括：标识与所述先前标识的缺陷相关联的一个或多个修复。

28.根据权利要求24所述的方法，其中，所述多个产物包括多个开发产物，并且还包括：基于字符、词语或短语在所述开发产物中的出现而从与所述聚类的多个产物相对应的所述多个开发产物中提取语义。

29.根据权利要求24所述的方法，其中，聚类所述多个产物包括：使用机器学习。

30.根据权利要求24所述的方法，其中，聚类所述多个产物包括：使用深度学习。

31.根据权利要求24所述的方法，其中，聚类所述多个产物包括：使用自动编码器。

32.根据权利要求24所述的方法，还包括：提供用于所述多个产物的所述聚类的训练，其中，所述训练包括使用软件文件的第一版本和所述软件文件的第二版本之间的一个或多个差异。

33.根据权利要求32所述的方法，其中，所述一个或多个差异与缺陷或修复相对应。

34.根据权利要求33所述的方法，其中，所述缺陷是安全漏洞，或所述修复是补丁。

35.根据权利要求32所述的方法，其中，所述一个或多个差异与特征或特征增强相对应。

36.一种用于标识设计模式的系统，包括：

一个或多个存储设备，具有用于多个文件中的每一个文件的多个产物；和

处理器，被配置成基于用于所述多个文件中的第一文件的所述多个产物中的至少一个产物来自动地标识设计模式。

37.根据权利要求36所述的系统，还包括：所述处理器还被配置成在所述第一文件中找到实现所述设计模式的程序片段。

38.根据权利要求36所述的系统，其中，自动地标识所述设计模式还包括：将所述设计模式的所述标识还基于用于所述多个文件中第二文件的所述多个产物中的至少一个产物，其中，所述第一文件和所述第二文件都属于项目。

39.根据权利要求36所述的系统，其中，所述设计模式是缺陷或修复。

40.根据权利要求36所述的系统，其中，所述设计模式是特征或特征增强。

41.根据权利要求36所述的系统，其中，所述设计模式是预先标识的程序片段。

42.一种用于标识设计模式的系统，包括：

一个或多个存储设备，具有多个产物；和

处理器，被配置成聚类所述多个产物，并且基于一个或多个先前标识的设计模式来从所述聚类中标识先前未标识的设计模式。

43.根据权利要求42所述的系统，其中，所述先前标识的设计模式是缺陷。

44.根据权利要求42所述的系统，还包括：标识与所述先前标识的缺陷相关联的一个或多个修复。

45.根据权利要求42所述的系统，其中，聚类所述多个产物包括：使用机器学习。

46.根据权利要求42所述的系统，其中，聚类所述多个产物包括：使用深度学习。

47.一种其上存储有可执行程序的非暂态计算机可读介质，其中，所述程序指示处理设备执行以下步骤：

访问具有用于多个文件中的每一个文件的多个产物的数据库；和

基于用于所述多个文件中的第一文件的所述多个产物中的至少一个产物来自动地标识设计模式。

48.一种其上存储有可执行程序的非暂态计算机可读介质，其中，所述程序指示处理设备执行以下步骤：

访问具有多个产物的数据库；

聚类所述多个产物；和

基于一个或多个先前标识的设计模式来从所述聚类中标识先前未标识的设计模式。

软件分析系统和方法

[0001] 相关申请

[0002] 本申请要求于2014年6月13日提交的美国临时申请第62/012,127号的权益。上述申请的全部教导通过引用并入本文。

[0003] 政府支持

[0004] 本发明是在来自美国空军的授权号FA8750-14-C-0056和来自国防高级研究计划局的授权号FA8750-15-C-0242的政府支持下进行的。政府对本发明具有一定的权利。

背景技术

[0005] 当今,软件开发、维护和修复是手动过程。软件供应商会计划、实施、记录、测试、部署和维护计算机程序。初始计划、实施、文档、测试和部署通常是不完整的,并且总是缺少所需的特征或包含缺陷。许多供应商都有生命周期维护计划来通过在软件成熟时推出迭代错误修复、安全补丁和功能增强来解决这些缺陷。

[0006] 世界上部署了大量的软件代码,数十亿行,并且维护和错误修复花费大量的时间和金钱来解决。历史上,软件维护一直是专门的和反应性的(即,响应于错误报告、安全漏洞报告和用户对特征增强的请求)手动过程。

发明内容

[0007] 本发明的实施例使软件开发、维护和修复生命周期的关键方面自动化,包括例如查找和修复程序缺陷,诸如错误(代码中的误差)、安全漏洞和协议缺失。本发明的示例实施例提供了可以利用包括公共可用的软件文件或专有软件在内的大量软件文件的系统和方法。

[0008] 根据本发明的一个实施例,一种用于标识设计模式的示例方法包括:访问具有用于多个文件中的每一个文件的多个产物的数据库,并且基于用于多个文件中的第一文件的多个产物中的至少一个产物来自动地标识设计模式。例如,文件可以是二进制代码格式、源代码格式或中间表示(IR)格式。

[0009] 对于某些实施例,设计模式在第一文件中。对于其它示例实施例,设计模式可以涉及诸如项目中的文件之类的文件或代码片段之间的交互,因此,自动地标识设计模式还可以基于用于第二文件等的产物。

[0010] 对于某些实施例,设计模式可以是缺陷、修复、特征、特征增强、或预先标识的程序片段。其它附加实施例可以位于多个产物中的至少一个产物中,诸如开发产物(developmental artifact)、表示缺陷的字符串、修复、特征、特征增强、或表示设计模式的预先标识的模式。用于示例实施例的产物可以是静态产物、动态产物、派生产物或元数据产物。

[0011] 附加的示例实施例可以将用于设计模式的标识符存储在数据库中。例如,可以使用用于设计模式的标签,诸如使用从用于第一文件的多个产物中的至少一个产物获得的字符串。附加的实施例可以在第一文件中找到与设计模式相对应的程序片段。

[0012] 根据本发明的一个实施例，一种用于标识设计模式的示例方法包括：访问具有与多个软件文件相对应的多个产物的数据库，以及通过自动分析与软件文件相关联的产物的至少一个产物来标识用于软件文件的至少一个软件文件的设计模式。示例方法的附加实施例还包括：将用于软件文件的软件模式的标识符存储在数据库中。

[0013] 对于某些示例实施例，产物包括以下各项中的一项或多项：直接插入代码注释、提交历史、文档文件、以及常见漏洞和暴露源条目。对于某些示例实施例，分析产物中的至少一个产物包括：在开发产物中搜索表示缺陷或修复的字符串。示例方法的附加的实施例还包括：在软件文件中找到实现设计模式的程序片段。对于某些示例实施例，通过在软件文件的中间表示中定位实现设计模式的代码来找到与设计模式相对应的程序片段。

[0014] 对于附加的示例实施例，存储用于软件文件的设计模式的标识符包括：使用从用于软件文件的产物中的一个或多个产物获得的字符串来存储用于设计模式的标签。对于示例实施例，设计模式是缺陷、修复、特征或特征增强。

[0015] 本发明的另一示例实施例是一种用于标识诸如缺陷之类的设计模式的方法，其包括：访问具有与软件文件相对应的产物的数据库，对产物进行聚类，以及基于一个或多个先前标识的设计模式从聚类来标识先前未标识的设计模式。对于某些示例实施例，设计模式是相同的，但是例如可以存在于另一文件中。对于某些示例实施例，示例方法还包括：标识与先前标识的缺陷相关联的一个或多个修复。

[0016] 对于某些示例实施例，产物包括开发产物，并且示例方法还包括：基于产物中的字符(包括字母数字或特殊字符)、单词或短语的出现从开发产物中提取语义。对于某些示例实施例，聚类多个产物包括：使用自动编码器。附加的实施例还包括：提供用于多个产物的聚类的训练，其中，训练包括使用软件文件的第一版本和软件文件的第二版本之间的一个或多个差异。对于某些实施例，这些差异可以与诸如安全漏洞之类的缺陷或诸如补丁之类的修复相对应。对于某些实施例，这些差异可以与特征或特征增强相对应。对于其它实施例，每种类型的产物被聚类。对于示例实施例，类型包括调用图、控制流图、使用定义链、定义使用链、支配树、基本块、变量、常量、分支语义、以及协议。对于某些示例实施例，聚类可以基于多种类型的产物。

[0017] 本发明的附加示例实施例是一种用于标识设计模式的系统，其包括：具有与软件文件相对应的产物的一个或多个存储设备，其中，产物包括存储在存储设备上的产物；以及处理器，其被配置成通过自动分析与软件文件相关联的产物中的至少一个产物来标识用于软件文件中的至少一个软件文件的设计模式。示例系统还可以具有被配置成在软件文件中找到实现设计模式的程序片段的处理器。

[0018] 本发明的附加示例实施例是一种用于标识设计模式的系统，其包括：具有多个产物的一个或多个存储设备；以及处理器，其被配置成聚类多个产物并且基于一个或多个先前标识的设计模式来从所述聚类中标识先前未标识的设计模式。对于某些示例性实施例，设计模式是缺陷、修复、特征、特征增强、或预先标识的模式。对于某些实施例，聚类包括使用机器学习或深度学习。

[0019] 本发明的附加示例实施例是其上存储有可执行程序的非暂态计算机可读介质，其中，该程序指示处理设备来执行以下步骤：访问具有与软件文件相对应的产物的数据库，以及基于用于多个文件中的第一文件的多个产物中的至少一个产物来自动地标识设计模式。

[0020] 本发明的附加的示例实施例是其上存储有可执行程序的非暂态计算机可读介质，其中，该程序指示处理设备来执行以下步骤：访问具有多个产物的数据库，对多个产物进行聚类，以及基于一个或多个先前标识的设计模式从所述聚类中标识先前未标识的设计模式。

附图说明

[0021] 从如下附图中所图示的本发明的示例性实施例的以下更具体的描述中，上述内容将是清楚的，其中在不同的视图中相同的附图标记是指相同的部件。附图不一定按比例绘制，重点在于图示了本发明的实施例。

[0022] 图1是图示了用于提供用于软件文件的语料库的方法的示例实施例的流程图。

[0023] 图2是图示了根据本发明的实施例的用于从用于语料库的输入软件文件提取中间表示(IR)的示例处理的流程图。

[0024] 图3是图示了根据本发明的实施例的用于软件文件的产物之间的分级关系的框图。

[0025] 图4是图示了用于提供软件文件的产物的语料库的系统的示例实施例的框图。

[0026] 图5是图示了用于标识设计模式的方法的示例实施例的框图。

[0027] 图6是图示了用于标识缺陷的方法的示例实施例的流程图。

[0028] 图7是图示了根据本发明的实施例的用于标识设计模式的产物的聚类的框图。

[0029] 图8是图示了用于使用语料库标识软件文件的方法的示例实施例的流程图。

[0030] 图9是图示了用于标识程序片段的方法的示例实施例的流程图。

[0031] 图10是图示了根据本发明实施例的使用语料库的系统的框图。

具体实施方式

[0032] 以下是本发明的示例实施例的描述。本文中所引用的任何专利或出版物的全部教导通过引用并入本文。

[0033] 根据本公开的示例实施例的软件分析允许从现有软件文件中利用知识，该现有软件包括来自公众可获得的源的或者是专有软件的文件。然后，该知识可以应用于其它软件文件，包括修复缺陷，标识漏洞，标识协议缺陷、或建议代码改进。

[0034] 本发明的示例实施例可以涉及软件分析的各个方面，包括创建，更新，维护或以其它方式提供软件文件的语料库和关于用于知识数据库的软件文件的相关产物。根据本发明的各方面，该语料库可以用于多种目的，包括自动地标识软件文件的更新版本，可用于软件文件的补丁，已知具有这些缺陷的文件中的缺陷，以及先前未知包含这些误差的文件中的已知缺陷。本发明的实施例还可以利用来自语料库的知识来解决这些问题。

[0035] 图1是图示了根据本发明的实施例的用于语料库的输入软件文件的示例处理的流程图。第一图示的步骤是获得多个软件文件110。这些软件文件可以采用源代码格式，其通常是纯文本，或者采用二进制代码格式或一些其它格式。进一步地，对于本发明的某些示例实施例，源代码格式可以是可以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、Python和Ruby。对于某些附加示例实施例，还可以获得用于与本发明的实施例一起使用的解释语言，包括PERL和bash脚本。

[0036] 所获得的软件文件不仅包括源代码或二进制文件，而且还可以包括与那些文件或对应软件项目相关联的任何文件。例如，软件文件还包括相关联的构建文件、构造文件、库、文档文件、提交日志、修订历史、bugzilla条目、通用漏洞和披露(CVE)条目、以及其它非结构化文本。

[0037] 可以从多种源获得软件文件。例如，可以经由因特网通过网络接口从诸如GitHUB、SourceForge、BitBucket、GoogleCode、或通用漏洞和披露系统之类的公众可获得的软件仓库(诸如由MITRE公司维护的公众可获得的软件仓库)来获得软件文件。通常，这些仓库包含文件和对文件所做改变的历史。还有，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。软件文件还可以经由接口从专用网络获得、或从本地硬盘驱动器或其它存储设备本地获得。接口提供用于通信地耦合到源。

[0038] 本发明的示例实施例可以获得可从源中获得的一些、大多数或所有文件。进一步地，一些示例实施例还自动获得文件，并且例如可以自动下载文件、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、目录中的所有文件、或可从源中获得的所有文件。一些实施例爬过整个存储库的每个修订以获得所有可用的软件文件。某些示例实施例获得用于语料库中的每个软件项目的整个源控制仓库，以便于自动获得项目的所有相关联的文件，包括获得每个软件文件修订。仓库的示例源控制系统包括Git、Mercurial、Subversion、并发版本系统、BitKeeper和Perforce。某些实施例还可以连续地或周期性地检查源以辨别源是否已经改变或更新，并且如果是，则可以仅从源获得改变或更新，或者也可以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，诸如示例实施例可以用于从源获得更新的添加日期或改变日期字段。

[0039] 本发明的某些示例实施例还可以单独获得库软件文件，该库软件文件可以由从仓库获得的源代码文件使用，以在仓库不包含库的情况下解决对这样的文件的需要。这些实施例中的某些实施例尝试获得可从任何公共源合理地获得或从软件供应商获得以包括在语料库中的任何库软件文件。附加地，某些实施例允许用户提供由软件文件使用的库或者标识所使用的库，以使可以获得它们。某些实施例刮除每个项目的软件文件以标识该项目所使用的库，以使如果需要，则可以获取并且安装它们。

[0040] 根据本发明的示例方法中的下一步骤是确定用于多个软件文件120中的每一个软件文件的多个产物。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、派生产物、以及元数据产物。

[0041] 示例方法的最后一步是将用于多个软件文件中的每一个软件文件的多个产物存储在数据库130中。多个产物以它们可以被标识为与从中确定它们的特定软件文件相对应的这样的方式而被存储。该标识可以以任何公知的多种方式来进行，诸如由数据库模式表示的数据库中的字段、指针、存储位置、或者诸如文件名之类的任何其它标识符。可以类似地跟踪属于同一项目或构建的文件，以使可以维护该关系。

[0042] 对于不同的实施例，数据库可以采取不同的形式，诸如图数据库、关系数据库或平面文件。一个优选实施例采用OrientDB，其是由Orient Technologies领导的OrientDB开方源项目提供的分布式图数据库。另一优选实施例采用Titan，其是为存储和查询分布在多机器群集上的图而优化的可扩展图数据库，以及Apache Cassandra存储后端。某些示例实施例还可以采用SciDB，其是来自范例4的还存储和操作图产物的阵列数据库。

[0043] 通常可以从源代码文件、二进制文件或其它产物来确定静态产物、动态产物、派生产物、以及元数据产物。在下文提供这些类型的产物的示例。示例实施例可以确定用于源代码或二进制软件文件的这些产物中的一个或多个产物。某些实施例不确定用于特定类型的这些类型的产物中的每一类产物或产物中的每一个产物，而是可以确定产物类型的子集和/或类型内的产物的子集，和/或根本没有特定类型。

[0044] 静态产物

[0045] 软件文件的静态产物包括调用图、控制流图、使用定义链、定义使用链、支配树、基本块、变量、常量、分支语义、以及协议。

[0046] 调用图(CG)是由函数调用的函数的有向图。CG表示高级程序结构并且被描绘为节点，其中，图的每个节点表示函数，并且节点之间的每个边缘是定向的并且示出了函数是否可以调用另一函数。

[0047] 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表示功能级程序结构。CFG中的每个节点表示基本块，节点之间的边缘是定向的，并且示出了流中的潜在路径。

[0048] 使用定义链(UD)和定义使用链(DU)是在基本代码块中执行的输入(使用)、输出(定义)和操作的有向非循环图。例如，UD链是变量的使用和可以达到该使用而无需干预重新定义的该变量的所有定义。DU链是变量的定义和可以从该定义达到而无需干预重新定义的所有使用。这些链使得能够对所接受的输入类型、所生成的输出类型和在基本代码块内部执行的操作的基本代码块进行语义分析。

[0049] 支配树(DT)是表示CFG中的哪些节点主导(处于路径中的)其它节点的矩阵。例如，如果从入口节点到第二节点的每个路径必须通过第一节点，则第一节点支配第二节点。DT以Pre(从前进)和Post(从后退)形式表达。当路径更改到CFG中的特定节点时，DT突出显示。

[0050] 基本块是CFG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。

[0051] 变量是用于信息及其类型的存储单元，表示对于任何函数参数、局部变量或全局变量可以存储的信息的类型，并且包括默认值(如果可用的话)。它们可以提供关于程序的初始状态和基本约束，并且示出了类型或初始值的改变，其可能会影响程序行为。

[0052] 常数是任何常数的类型和值，并且可以提供关于程序的初始状态和基本约束。它们可以示出了类型或初始值的改变，其可能会影响程序行为。

[0053] 分支语义是if语句和循环内部的布尔估计。分支对执行它们的基本块的条件进行控制。

[0054] 协议是协议、库、系统调用和程序所使用的其它已知函数的名称和引用。

[0055] 本发明的示例实施例可以从软件源代码文件的中间表示(IR)自动确定静态产物，诸如由公众可获得的LLVM(以前称为低级虚拟机)编译器基础架构项目提供的静态产物。LLVM IR是一种低级公共语言，其可以有效地表示高级语言，并且独立于指令集架构(ISA)，诸如ARM、X86、X64、MIPS和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)来将源代码转换为公共LLVM IR。至少Ada、C/C++、D、Erlang、Haskell、Java、Lua、Objective C/C++、PHP、Pure、Python和Ruby的前端是公众可获得的。进一步地，可以容易地编程用于附加语言的前端。LLVM还有可用的优化器，以及可以将LLVM IR转换为用于多种不

同ISA的机器语言的后端。附加的示例实施例可以从源代码文件来确定静态产物。

[0056] 图2是图示了可以根据本发明的实施例利用的语料库的输入软件文件的附加示例处理的流程图。除了其它方面，示例实施例可以获得源代码205和二进制代码210软件文件。当LLVM编译器220可用于源代码文件205的语言时，用于该语言的LLVM编译器220可以用于将源代码翻译成LLVM IR 250。对于没有用可用LLVM编译器编译的语言，源代码205可以首先用用于该语言的任何支持的编译器215被编译成二进制文件230。然后，使用诸如Fracture之类的反编译器235来反编译二进制文件230，该反编译器是由Draper Laboratory提供的公众可获得的开放源反编译器。反编译器235将机器代码230翻译成LLVM IR 250。对于以二进制形式210获得的文件(其是机器代码230)，使用反编译器235对它们进行反编译以获得LLVM IR250。示例实施例可以从LLVM IR提取与语言无关的产物和与ISA无关的产物。

[0057] 本发明的示例实施例可以自动获得用于源代码软件文件中的每个源代码软件文件的IR。例如，示例实施例可以自动在仓库中搜索用于标准构建文件(诸如autocomf、cmake、automake或makefile或供应商指令)的项目。示例实施例可以通过监视构建过程并且将编译器调用转换为针对源代码的特定语言的LLVM前端调用来自动选择性地尝试使用这样的文件来构建项目。用于构建文件的选择过程可以逐步通过文件中的每个文件以确定哪个文件存在并且提供完成的构建或部分完成的构建。

[0058] 附加的示例实施例可以在从仓库中自动获得文件、将文件转换为LLVM IR、和/或确定用于文件的产物中使用分布式计算机系统。示例分布式系统可以使用主计算机来推送项目并且构建到从属计算机进行处理。从属可以各自处理它们被分配的项目、版本、修订或构建，并且可以将源或二进制文件翻译为LLVM IR，和/或确定产物并且提供用于在语料库中存储的结果。某些示例实施例可以采用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放源软件框架。从源仓库获得文件也可以分布在一组机器之间。

[0059] 根据示例性实施例，软件文件和LLVM IR还可以存储在语料库中，包括在分布式存储中。示例实施例还可以确定软件文件或LLVM IR代码已经存储在数据库中并且选择不再次存储文件。指针、图形数据库中的边缘或其它引用标识符可以用于将文件与特定项目、目录或其它文件集合相关联。

[0060] 动态产物

[0061] 动态产物表示程序行为，并且通过在诸如虚拟机、仿真器(例如，快速仿真器(“QEMU”)或管理程序之类的仪器化环境中运行软件来生成。动态产物包括系统调用跟踪/库跟踪和执行跟踪。

[0062] 系统调用跟踪或库跟踪是执行系统调用或库调用的顺序和频率。系统调用是程序如何从操作系统的内核中请求服务，该内核管理输入/输出请求。库调用是对软件库的调用，该软件库是可以重新用于开发软件程序和应用程序的编程代码的集合。

[0063] 执行追踪是包括指令字节、堆栈帧、存储器使用(例如，驻留/工作集大小)、用户/内核时间和其它运行时间信息的每指令跟踪。

[0064] 本发明的示例实施例可以产生虚拟环境，包括用于多种操作系统，并且可以运行并且编译源代码和二进制文件。这些环境可以允许确定动态产物。例如，可以采用诸如Valgrind或Daikon之类的公众可获得的程序来提供关于程序的运行时间信息以用作产物。

Valgrind是特别用于调试内存、检测内存泄漏和剖析的工具。Daikon是可以在代码中检测不变量的程序;不变量是在代码中的某些点处成立的条件。

[0065] 其它实施例可以使用公众可获得的附加诊断和调试程序或实用程序,诸如strace和dtrace。Strace用于监视进程和内核之间的交互,包括系统调用。Dtrace可以用于为系统提供运行时间信息,包括所使用的内存量、CPU时间、特定函数调用、以及访问特定文件的进程。示例实施例还可以在程序的多个运行中跟踪执行跟踪(例如,使用Valgrind)。

[0066] 附加的实施例可以通过KLEE引擎来运行LLVM IR。KLEE是符号虚拟机,其是公众可获得的开放源代码。KLEE符号地执行LLVM IR并且自动生成执行所有代码程序路径的测试。符号执行特别涉及分析代码以确定什么输入使得执行代码的每个部分。采用KLEE在发现功能正确性误差和行为不一致性方面非常有效,并且因此允许本发明的示例实施例快速标识类似代码中的差异(例如,跨修订)。

[0067] 派生产物

[0068] 派生产物表示复杂的高级程序行为并且提取这些行为的特点的属性和事实。派生产物包括程序特点、循环不变量、扩展类型信息、Z符号、以及标签转换系统表示。

[0069] 程序特点是关于从执行跟踪中导出的程序的事实。这些事实包括最小、最大和平均内存大小;执行时间;和堆栈深度。

[0070] 循环不变量是在循环的所有迭代(或选定的迭代组)上维护的属性。循环不变量可以映射到分支语义以发现类似行为。

[0071] 扩展类型信息包括关于类型的事实,包括变量可以保存的值的范围、与其它变量的关系、以及可以被抽象的其它特征。类型约束可以揭示关于代码的行为和特征。

[0072] Z符号基于Zermelo-Fraenkel集合理论。它提供了类型代数符号,从而使得能够比较基本块和忽略结构、顺序和类型的整个函数之间的度量。

[0073] 标签转换系统(LTS)表示是表示从程序抽象的高级状态的图系统。图的节点是状态,并且边缘由转换中的相关联的动作来标记。

[0074] 对于某些示例实施例,可以从其它产物、从包括使用上文针对动态产物所描述的程序的源代码文件、以及从LLVM IR来确定派生产物。

[0075] 元数据产物

[0076] 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。这些产物与计算机程序具有上下文关系。元数据产物包括文件名、版本号、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。元数据产物的子集可以被称为开发产物,其是与文件、程序或项目的开发过程有关的产物。开发产物可以包括直接插入代码注释、提交历史、bugzilla条目、CVE条目、构建信息、配置脚本、以及文档文件,诸如README.*TODO.*。

[0077] 示例实施例可以采用Doxygen,其是可公开获得的文档生成器。Doxygen可以从特别注释的源代码文件(即,直接插入代码文档)生成用于程序员和/或最终用户的软件文档。

[0078] 附加的实施例可以使用解析器,诸如另一语言识别(ANTLR)4生成的解析器工具,以产生抽象语法树(AST)以提取高级语言特征,其还可以用作产物。ANTLR4采用语法、语言的字符串的生成规则,并且生成可以构建和行走解析树的解析器。结果解析器发出各种类型、函数定义/调用、以及与程序结构有关的其它数据。用ANTLR4生成的解析器提取的低级属性包括复杂类型/结构、循环不变量/计数器(例如,来自用于每个范例的a)和结构化注释

(例如,正式前/后条件语句)。因为文件名、行和列号信息存在于解析器和LLVM IR中,所以示例实施例可以将这个提取的数据映射到其在LLVM IR中的参照位置。

[0079] 本发明的示例实施例可以通过从源软件文件提取字符串(诸如直接插入注释)来自动确定一个或多个元数据产物。其它实施例从文件系统或源控制系统自动确定元数据产物。

[0080] 分级产物间关系

[0081] 图3是图示了根据本发明的实施例的用于软件文件的产物之间的分级关系的方框图。示例实施例可以维护并且利用这些分级产物间关系。进一步地,不同的实施例可以使用不同的模式和不同的分级关系。对于图3的示例实施例,产物层级的顶部是LTS产物310。每个LTS节点310可以映射到功能和特定可变状态的集合或子集。在LTS产物310下是CG产物320。每个CG节点320可以利用CFG产物330映射到特定函数,其边缘可以包含循环不变量和分支语义330。每个CFG节点330可以包含基本块和DT 340。在那些产物下面是变量、常数、UD/DU链和IR指令350。图3清楚地图示了产物可以从描述动态信息范围的LTS节点低到单独的IR指令而映射到分级结构的不同级别。这些分级关系可以由用于多种用途的示例实施例使用,包括诸如通过首先比较更靠近层级的顶部的产物(与更接近底部的产物相比较)来更有效地搜索匹配产物,以便取决于较高级产物是否是匹配而包括或排除与较高级产物相关联的较低级产物的整个集合。附加的实施例还可以在定位或建议修复代码中利用分级关系用于缺陷或特征增强,包括通过在层级中更高来定位具有匹配的更高级产物的缺陷的修复代码。

[0082] 图4是图示了用于提供用于软件文件的产物的语料库的系统的示例实施例的方框图。示例实施例可以具有能够与具有多个软件文件的源430进行通信的接口420。对于某些实施例,该接口420可以通信地耦合到本地源430,诸如本地硬盘驱动器或盘。在其它实施例中,接口420可以是用于通过公共或专用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、或通用漏洞和披露系统。私人源的示例包括公司的内部网络和存储在其上的文件,包括在共享网络驱动器和私人仓库中。该示例系统还具有耦合到接口420以从源430获得多个软件文件的一个或多个处理器410。处理器410还可以用于确定用于多个软件文件中的每一个软件文件的多个产物。这些产物可以是静态产物、动态产物、派生产物和/或元数据产物。对于附加的实施例,处理器410还可以被配置成将软件文件中的每个软件文件转换成中间表示并且从中间表示中确定产物。

[0083] 示例系统还具有用于存储软件文件中的每一个软件文件的产物的一个或多个存储设备440a-440n,并且耦合到处理器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和分布式存储,诸如通过采用Hadoop文件系统(HDFS)上的Titan和Cassandra提供的。同样,示例系统可以具有一个处理器410或者采用分布处理并且具有多于一个的处理器410。另外的实施例还提供了接口420和存储设备440a-440n之间的直接通信耦合。

[0084] 图5是图示了用于定位设计模式的方法的示例实施例的方框图。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层级的各个级别处提取的产物(例如,规范、CG、CFG、定义使用链、指令序列、

类型和常量)相关联。

[0085] 示例方法提供访问具有与多个软件文件510相对应的多个产物的数据库。数据库可以是图数据库、关系数据库或平面文件。数据库可以位于本地,在专用网络上,或经由因特网或云而可用。一旦已经访问了数据库,则该方法可以基于用于多个文件520中的第一文件的多个产物中的至少一个产物来自动地标识设计模式。对于某些示例实施例,多个产物中的每一个产物可以是静态产物、动态产物、派生产物或元数据产物。其它实施例可以具有不同类型的产物的混合。进一步地,文件的格式不受限制,并且例如可以是二进制码格式、源代码格式或中间表示(IR)格式。

[0086] 对于某些实施例,设计模式可以通过关键字搜索或开发产物的自然语言搜索来标识。例如,源代码文件的修订中的直接插入代码注释可以标识所找到并且修复的缺陷。注释可以使用诸如缺陷、错误、误差、问题、缺点或毛刺(glitch)之类的词语。这些词语可以用于元数据的关键字搜索。提交日志还可以包括描述为什么应用新修订和补丁的文本,诸如解决缺陷或增强特征。进一步地,可以将训练和反馈应用于搜索以改进搜索努力。

[0087] 附加的示例实施例可以从CVE源来搜索开发产物,其标识文本中的常见漏洞和误差,并且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获得并且存储在数据库中。某些源还编码缺陷,以使代码可以用作关键字来定位哪个文件包含缺陷。附加地,可以在软件文件的标识中考虑和加权产物的源。例如,CVE源在标识缺陷方面可能比没有原产地或直接插入注释的仓库更可靠。其它实施例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标识软件文件,并且基于匹配附加产物(诸如例如,CG或CFG)来确认该标识。

[0088] 本发明的某些实施例执行示例方法,并且尝试标识一些、大多数或所有源代码和LLVM IR文件的设计模式。附加地,每当将文件添加到语料库时,某些实施例访问数据库并且尝试标识任何设计模式。某些实施例还可以标记所标识的设计模式以供稍后使用。

[0089] 某些实施例还找到源代码或与也已经存储在数据库中的文件相关联的LLVM IR中的缺陷的位置。例如,开发产物可以指定源代码中缺陷存在的位置以及补丁中修复存在的位置。还有,可以分析源代码或LLVM IR,并且与具有缺陷和文件的较新修复版本的文件进行比较,以隔离差异并且辨别缺陷和修复所在的位置。对于某些实施例,在开发产物中标识的缺陷类型还可以用于缩小对代码中缺陷位置的搜索。附加的实施例还可以标识设计模式,诸如使用标签,并且将该标识符存储在该文件的数据库中。这允许数据库容易地搜索某些缺陷或缺陷的类型。这样的标签的示例包括从用于软件文件的开发产物或从源代码获得的字符串。这种相同的途径可以应用于标识特征和特征增强并且标记它们。

[0090] 对于某些示例实施例,设计模式位于软件文件中。对于某些示例实施例,设计模式可以与文件之间的交互(诸如接口)有关。示例实施例可以通过将标识基于用于多个软件文件(诸如都属于软件项目的第一文件和第二文件)的产物来自动地标识设计模式。例如,表示设计模式(诸如接口不匹配误差)的预先标识的模式可以存储在数据库或允许来自第一文件和第二文件的产物用于标识对于这些文件来说存在接口误差的别处。示例实施例的示例设计模式包括缺陷、修复、特征、特征增强、或预先标识的程序片段。

[0091] 对于某些示例实施例,该方法在产物中定位表示缺陷或修复的字符串。通常,这样的字符串(诸如错误、误差或缺陷)存在于开发产物中,以及关于修复的字符串以及在代码

中可以找到的字符串。这些开发产物还可以具有表示特征或特征增强的字符串。

[0092] 对于某些示例实施例，设计模式基于表示设计模式的预先标识的模式。这些预先标识的模式可以由用户创建，可以通过与本公开相关联的方法来预先标识，或者可以以某种其它方式来标识。这些预先标识的模式可以与缺陷、修复、特征、特征增强、或感兴趣的项目或其它显著性相对应。

[0093] 图6是图示了用于定位缺陷的方法的示例实施例的流程图。该方法包括访问具有与多个软件文件相对应的多个软件产物的数据库610(诸如语料库)。然后，分析产物以从数据量中标识模式。例如，该分析可以包括：对多个产物620进行聚类。通过对数据进行聚类，可以找到未知包含已知缺陷的文件中的已知缺陷。因此，根据聚类，示例方法可以基于一个或多个先前标识的缺陷630来标识先前未标识的缺陷。

[0094] 本发明的某些示例实施例可以对语料库采用机器学习。机器学习涉及通过从低级产物开始来捕获数据中的相关特征，然后建立更复杂的表示来学习数据的分级结构。某些示例实施例可以对语料库采用深度学习。深度学习是基于数据的学习表示的更广泛的机器学习方法族的子集。对于某些实施例，自动编码器可以用于聚类。

[0095] 对于某些示例实施例，产物可以由一组自动编码器来处理，以自动发现未标记图的紧凑表示和文档产物。图产物包括可以以图形式表达的那些产物，诸如CG、CFG、UD链、DU链和DT。然后可以对图产物的紧凑表示进行聚类以发现软件设计模式。从对应的元数据产物提取的知识可以用于标记设计模式(例如，错误、修复、漏洞、安全补丁、协议、协议扩展、特征、以及特征增强)。

[0096] 对于某些示例实施例，自动编码器是结构化稀疏自动编码器(SSAE)，其可以将向量作为输入并且提取公共特征。对于某些实施例，为了自动发现程序的特征，首先以矩阵形式表达所提取的图形产物。许多所提取的产物可以表达为邻接矩阵，包括例如，CFG、UD链和DU链。可以在软件文件和项目层级的每个级别来学习结构特征。

[0097] 图产物中节点的数目可以宽泛地变化；因此，可以提供中间产物作为深度学习的输入。一个这样的中间产物是图拉普拉斯算子的第一k个本征值，使得深度学习能够执行类似于频谱聚类的处理。其它中间产物包括聚类系数，提供图中节点趋向于聚类在一起的程度的度量，诸如全局聚类系数、网络平均聚类系数和传递率。另一中间产物是图的荫度、图的密度的度量。具有许多边缘的图具有高荫度，并且具有高荫度的图具有密集子图。另一个中间产物是等差数，即，图是否具有瓶颈的数值度量。这些中间产物捕获用于机器学习方法的图的结构的不同方面。

[0098] 机器学习，包括深度学习，例如，实施例可以采用使用从简单的自动编码器结构开始的多步骤过程来训练的算法，并且迭代地细化该途径以开发SSAE。SSAE还可以被训练以从中间产物中学习特征。自动编码器学习未标记数据的紧凑表示。它可以通过神经网络来建模，该神经网络由至少一个隐藏层组成并且具有相同数目个输入和输出，其学习身份函数的近似。自动编码器将输入信号脱水(dehydrate)(编码)为一组基本的描述性参数，并且对那些信号进行再水合(rehydrate)(解码)以重新创建原始信号。可以在训练期间自动选择描述性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将信号分组成聚类的基础。

[0099] 自动编码器可以通过将输入信号映射到较低维度特征空间来降低这些输入信号

的维度。示例实施例然后可以对由自动编码器发现的特征空间中的代码执行聚类和分类。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，其将特征分成k个聚类，其使得所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k的初始数目。因为k均值聚类的操作度量基于欧几里得距离，所以搜索潜在聚类的数目，从而为许多不同k的每一个k计算新结果是非常有效的。示例实施例可以使用在从其导出聚类特征的软件文件内最频繁出现的主题的标签来分类所得到的群集。

[0100] 尽管特征向量是稀疏和紧凑的，但是可能难以仅通过检查特征向量来理解输入向量。因此，示例实施例可以利用与先前学习的权重参数相关联的先验。给定足够的语料库，对于“修复”代码应当例如出现参数空间中的模式。示例实施例可以使用由直到该点收集的数据集给出的先验信息将特定模式并入自动编码器中。特别地，当标签由系统学习时，示例实施例可以将该信息并入自动编码器操作中。

[0101] 示例实施例可以使用数据库管理(例如，联接、滤波)和分析操作(例如，奇异值分解(SVD)、双聚类)的混合。示例实施例的图理论(例如，谱聚类)和机器学习或深度学习算法都可以使用类似的算法基元以用于特征提取。SVD还可以用于对用于学习算法的输入数据进行去噪，并且使用较少的维度来近似数据，并且因此执行数据简化。

[0102] 示例实施例可以通过包括经由文本分析的文档产物的无监督的语义标签生成来封装人随着时间的推移并且跨程序的代码状态的理解。文本分析的示例是隐含狄利克雷分配(LDA)。可以使用LDA和主题建模从文档产物中提取语义信息。这些途径是“词袋(bag-of-words)”技术，其查看词语或短语的出现，忽略顺序。例如，表示“科学计算”的袋子可以具有诸如“FFT”、“小波”、“sin”和“atan”的种子术语。示例实施例可以使用来自源的所提取的文档产物，诸如源注释、CG/CFG节点标签和提交消息，以通过计数术语的出现来填充“袋子”。所得到的固定仓直方图可以被馈送到受限玻尔兹曼机(RBM)，其是适于文本应用的深度学习算法的实现方式。所提取的主题捕获与所提取的文档产物相关联的语义信息，并且可以用作经由自动编码器由图产物的无人监督学习形成的聚类的标签(例如，错误/修正、漏洞/补丁)。可以由附加示例实施例采用的其它形式的文本分析包括自然语言处理、词汇分析和预测分析。

[0103] 从文档产物提取的主题标签可以提供标记信息以通知自动编码器的结构化。示例实施例可以基于学习过的主体、表示顺序软件模式(即，在软件修订之前/之后)的语义共性来查询语料库数据库的训练数据群体。这些模式可以捕获嵌入软件开发文件(诸如提交日志、改变日志和注释)中的改变，其随时间而与软件开发生命周期相关联。这些改变的关联提供了对与检测和修复相关的软件的演变的洞悉，诸如错误/修正、漏洞/安全补丁、以及特征/增强。该信息还可以用于理解和标记从产物语料库自动提取的知识。

[0104] 图7示出了图示了根据本发明的实施例的用于标识设计模式的产物的聚类的方框图。可以在软件文件层级的每个级别(包括系统、程序、功能和方框710)学习结构特征。可以针对聚类715分析诸如CG、CFG和DT之类的图产物。这些图产物可以变换成图不变量特征720。然后可以将这些图特征740提供为图分析模块760(诸如自动编码器)的输入，并且针对类似的设计模式来审查所得到的聚类，其被聚类在一起780。文本(诸如来自源代码文件或来自开发产物的一个或多个字符串)可以被映射到标签730。这些标签750可以诸如通过使用LDA或其它自然语言处理而由文本分析模块770来分析，并且标签可以与从其导出标签的

对应发现的聚类780相关联。这些模块760，770可以以软件、硬件或其组合来实现。

[0105] 图8示出了图示了用于使用语料库来标识软件的方法的示例实施例的流程图。该示例实施例获得软件文件810。该文件可以经由因特网、云或私人公司的服务器而从公共或私人源(诸如公共仓库)经由网络接口来获得。某些示例实施例还可以从本地源(诸如本地硬盘驱动器、便携式硬盘驱动器、或盘)获得软件文件。示例实施例可以从源获得单个文件或多个文件，并且可以诸如经由使用脚本语言或者通过用户交互手动地自动地这样做。示例方法然后可以确定用于软件文件的多个产物820，诸如本文中所描述的任何其它产物。示例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参照软件文件的多个参照产物。参照产物可以存储在语料库数据库中。对于某些示例性实施例，这些参照文件可以包括先前已经获得的并且其产物已经与用于某些实施例的软件文件一起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中的参照产物或其多个子集进行比较840。示例实施例可以通过标识具有匹配多个产物的多个参照产物的参照软件文件来标识软件文件850。因为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标识为是相同的文件。

[0106] 然后，还可以比较附加的产物或代码部分以增加进行正确标识的置信水平。置信度可以是固定的或可调整的，并且可以基于广泛多种的标准，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，可以对特定数据集及其观察进行这种调整。更进一步地，对于某些实施例，匹配可以包括模糊匹配，诸如具有小于100％匹配的百分比的可调设置，以具有所声明的匹配。

[0107] 对于某些示例实施例，在匹配和标识过程中可以给予某些产物更多或更少的权重。例如，诸如指令是否与32位或64位处理器相关联之类的常见产物可以被给予零的权重或一些其它较小权重。一些产物在变换下可以是或多或少不变的，并且对于某些示例实施例，可以相应地调整用于这些产物的权重。例如，文件名或CG产物可以被认为在建立文件的身份方面是高信息性的，而某些产物(诸如LTS或DT)例如可以被认为是较少指示性的，并且对于某些示例性实施例和源给予较小的权重。附加的实施例可以给予产物的某些组合更大的权重，以在进行比较时标识匹配。例如，使得CFG和CG产物匹配可以在进行标识时比使得基本块产物和DT产物匹配给予更多的权重。同样，在进行文件的标识时，可以给予不匹配的某些产物更多或更少的权重。在标识过程中评估加权的附加的示例可以包括表达标识阈值，诸如以匹配产物的百分比或一些其它度量。附加的实施例可以更改标识阈值，包括基于诸如文件的源、文件的类型、时间戳(包括文件的日期、文件的大小、或者该文件的某些产物是否不能确定或以其它方式不可用)。

[0108] 附加实施例可以通过将软件文件转换为诸如LLVM IR之类的中间表示，并且从中间表示确定多个产物中的至少一个产物来确定用于软件文件的多个产物中的一些产物。其它实施例可以通过从软件文件中提取字符串来确定多个产物中的一些产物，诸如源代码文件或文档文件。

[0109] 示例实施例还可以包括通过分析与所标识的参照软件文件相关联的至少一个参照产物来确定是否存在软件文件的较新版本。例如，一旦已经标识了软件文件，就可以检查数据库以查看软件文件的较新修订是否可用，诸如通过检查对应的参照文件的修订号或时间戳，或者可以将参照文件标识为另一文件的旧版本的与数据库中的产物和文件相关联的

标签。附加的示例实施例还可以自动提供软件文件的较新版本，包括给用户或公众或私人源。

[0110] 某些附加实施例可以通过分析与所标识的参照软件文件相关联的至少一个参照产物来确定软件文件的补丁是否存在。例如，示例实施例可以检查与参照软件文件相关联的产物，并且确定文件存在补丁，包括尚未应用于软件文件的补丁。附加实施例可以自动地将补丁应用于软件文件或者提示用户他们是否想要应用补丁。

[0111] 某些附加实施例可以针对某些实施例分析补丁以及软件文件(或者参照软件文件，因为它们匹配)，以确定与软件文件中的缺陷的修复相对应的补丁的修复部分。该分析可以在针对某些实施例获得软件文件之前或之后发生。附加的实施例可以仅将补丁的修复部分应用于软件文件，包括自动地或提示用户他们是否应用补丁的修复部分。附加的实施例可以将补丁的修复部分提供给源以在源处应用该修复部分。进一步地，补丁和软件文件的分析可以包括将补丁和软件文件转换为中间表示，并且从中间表示确定多个产物中的至少一个产物。类似地，附加的实施例可以分析补丁和软件文件(或者参照软件文件，因为它们匹配)，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。附加的实施例可以仅将补丁的特征增强部分应用于软件文件，包括自动地或提示用户他们是否希望应用补丁的特征增强部分。

[0112] 附加的示例实施例可以通过分析与所标识的参照软件文件相关联的至少一个参照产物来确定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标识为具有修复可用的缺陷的产物。附加的实施例可以自动修复软件文件中的缺陷，包括通过用源代码的修复块替换源代码的块、或用中间表示的修复块替换软件文件中的中间表示的块。附加的实施例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。对于某些实施例，所修复的文件可以被发送到软件文件的源。附加的实施例可以提供修复代码以提供给软件文件的源，以便在那里修复文件。

[0113] 图9是图示了用于标识代码的方法的示例实施例的流程图。该示例方法可以获得一个或多个软件文件910。对于软件文件，可以确定多个产物920。如果产物已经被确定的话，则某些实施例可以替代地获得产物，而非确定产物。可以访问存储多个参照产物的数据库930。参照产物是如本文中所描述的产物，并且可以与参照软件文件、参照设计模式或感兴趣的代码的其它块相对应。数据库可以存储在许多位置，诸如本地存储，或者存储在网络驱动器上，或者可以通过互联网或在云中访问，并且还可以分布在多个存储设备上。然后，可以通过将与程序片段相对应的多个产物和与该程序片段相对应的多个参照产物进行匹配来标识在一个或多个软件文件中或与它们(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、功能或功能之间的接口的子部分。程序片段可以与单个指令一样小，或者与整个文件、程序、基本块、功能或接口一样大。所选择的部分可以足以以任何期望的置信度来标识程序片段，其对于某些实施例可以是可设置或可调整的，并且可以更改，诸如上文关于标识文件所描述的。

[0114] 对于某些实施例，确定用于软件文件的产物包括将软件文件转换为中间表示，并且从中间表示确定至少一个产物。对于某些实施例，软件文件和参照软件文件各自采用源代码格式或各自采用二进制代码格式。对于附加的实施例，程序片段与软件文件中的缺陷相对应并且已经在数据库中被标识以与缺陷相对应。附加的实施例可以自动地修复软件文

件中的缺陷或向用户提供一个或多个修复选项以修复缺陷。某些实施例可以订购修理选项，包括例如基于由用户选择的一个或多个先前修复选项或基于用于修复选项的成功可能性。

[0115] 图10是图示了根据本发明的实施例的使用软件文件的数据库语料库的系统的方框图。示例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还通信地耦合到处理器1030。对于附加的实施例，接口1020还可以直接耦合到存储设备1040。该存储设备1040可以是广泛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可以存储包括用于多个参照软件文件中的每一个参照软件文件的参照产物，并且可以通信地耦合到处理器1030。处理器1030可以被配置成使得从源1010获得软件文件。该软件文件的身份以及是否有该文件的更新版本可用，是否有补丁可用，或该文件是否包含缺陷或未增强特征都是示例系统可以解决的问题的示例。处理器1030还被配置成确定软件文件的多个产物，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中的参照产物进行比较，并且通过标识具有与用于软件文件的比较的产物相对应的参照产物的参照软件文件来标识软件文件。

[0116] 在示例系统的附加的实施例中，处理器1030可以被配置成如果在文件的存储设备1040中可用，则自动向软件文件应用补丁。在另外的实施例中，处理器还可以被配置成分析所标识的补丁和软件文件以确定是否存在补丁的修复部分，其与软件文件中的缺陷的修复相对应，并且如果是，仅将补丁的修复部分应用于软件文件，或提示用户。

[0117] 图10的框图还可以图示了根据本发明的实施例的使用数据库语料库的另一示例系统。该另一个图示的示例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还通信地耦合到处理器1030。对于附加的实施例，接口1020还可以直接耦合到存储设备1040。该存储设备1040可以是广泛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，并且可以通信地耦合到处理器1030。处理器1030可以被配置成使得获得一个或多个软件文件，以确定用于一个或多个软件文件的多个产物，访问存储多个参照产物的数据库，以及通过将与程序片段相对应的多个产物和与程序片段相对应的多个参照产物进行匹配来标识用于一个或多个软件文件的程序片段。对于某些示例实施例，在数据库中已经标识出程序片段以与缺陷相对应。这种缺陷的示例包括错误、安全漏洞和协议缺点。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有关。附加的实施例还可以使处理器被配置成自动修复一个或多个软件文件中的缺陷。对于某些示例实施例，在数据库中已经标识出程序片段以与特征相对应，并且某些实施例还可以自动地提供特征增强，包括采用源代码或二进制文件的补丁的形式。

[0118] 修复

[0119] 示例实施例支持用于自动修复的程序合成，包括通过替换CG节点(函数)、CFG节点(基本块)、特定指令、或特定变量和常数来实例化所选择的修复。这些元素(例如，函数、基本块、指令)可与具有兼容接口(即，相同数目个参数、类型和输出)的元件交换，并且可以通过将LLVM IR的缺陷块替换为LLVM IR的修复块来变换LLVM IR。

[0120] 某些实施例还可以选择用函数调用和具有一个或多个基本块的函数调用来交换

基本块。某些实施例可以修补源代码和二进制。当它们不存在时，附加实施例还可以创建用于交换的合适元素。高级产物(例如，LTS和Z谓词)可以用于导出用于软件补丁的兼容实现方式。示例实施例可以利用所提取的图表示的层级，首先将层级升序到修复模式的合适表示，然后(经由编译)层级降序到具体实现方式。产物的分级性质可以帮助形成修复代码。

[0121] 示例实施例可以允许用户提交目标程序(源或二进制)，并且示例实施例发现任何缺陷设计模式的存在。对于每个缺陷，可以向用户提供候选修复策略(即，修复设计模式)。用户可以选择要合成的修复和要修补的目标的策略。某些示例实施例还可以从用户选择中学习以最佳地排序未来的修复解决方案，并且修复策略也可以按排序的顺序来呈现给用户。某些实施例还可以自动运行，修复整个软件语料库(包括连续地、周期性地和/或在设计环境中)的缺陷或漏洞。

[0122] 除了上文所讨论的实施例之外，本发明可以用于广泛多种用途。例如，在软件代码的编程期间可以使用示例实施例来辅助程序员，包括标识缺陷或建议代码重用。附加的示例实施例可以用于发现缺陷和漏洞并且可选地自动修复它们。另一其它示例实施例可以用于优化代码，包括标识未使用的代码、低效代码和用于替换效率较低的代码的建议代码。

[0123] 示例实施例还可以用于风险管理和评估，包括关于在某些代码中可能存在什么漏洞。附加的实施例还可以用在设计认证过程中，包括提供软件文件没有已知缺陷(诸如错误、安全漏洞和协议缺陷)的证明。

[0124] 本发明的又一其它附加的示例实施例包括：代码重用发现器(找到已经在代码库中执行相同事情的代码)、代码质量测量、对代码翻译器的文本描述、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/误差估计器、无用代码发现、代码特征映射、自动补丁审阅器、代码改进决策工具(特征列表到最小改变的映射)、对现有设计工具(例如，企业架构师)的扩展、备选实现建议器、代码探索和学习工具(例如，用于教学)、系统级代码许可证足迹、以及企业软件使用映射。

[0125] 应当理解，上文所描述的示例实施例可以以许多不同的方式来实现。在一些实例中，本文中所描述的各种方法和机器可以各自由具有中央处理器、存储器、盘或其它大容量存储装置、(多个)通信接口、(多个)输入/输出(I/O)设备和其它外围设备的物理、虚拟或混合通用计算机来实现。通用计算机被变换成执行上文所描述的方法的机器，例如，通过将软件指令加载到数据处理器中，然后使指令的执行来执行本文中所述的功能。软件指令还可以被模块化，诸如具有用于摄取文件以形成语料库的摄取模块、用于确定用于语料库的文件的产物和/或待被标识或分析用于设计模式的文件的分析模块、图分析模块、以及用于执行机器学习的文本分析模块、用于标识文件或设计模式的标识模块、以及用于修复代码或提供更新或修复的文件的修复模块。对于某些示例性实施例，这些模块可以组合或分离成附加模块。

[0126] 如本领域中已知的，这样的计算机可以包含系统总线，其中，总线是用于在计算机或处理系统的组件之间进行数据传送的一组硬件线。总线或多个总线基本上是连接计算机系统的不同元件(例如，处理器、磁盘存储装置、存储器、输入/输出端口、网络端口等)的共享管道，其使得能够在元件之间传送信息。一个或多个中央处理器单元附接到系统总线并且提供计算机指令的执行。还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。(多个)网络接口允许

计算机连接到附接到网络的各种其它设备。存储器为用于实现实施例的计算机软件指令和数据提供易失性存储。磁盘或其它大容量存储为用于实现例如本文中所描述的各种进程的计算机软件指令和数据提供非易失性存储。

[0127] 因此，实施例通常可以在硬件、固件、软件或其任何组合中实现。更进一步地，示例实施例可以完全或部分地驻留在云上，并且可以经由因特网或其它网络架构来访问。

[0128] 在某些实施例中，本文中所描述的进程、设备和过程构成计算机程序产品，包括提供用于系统的软件指令的至少一部分的非暂态计算机可读介质，例如，可移除存储介质，诸如一个或多个DVD-ROM、CD-ROM、磁盘、磁带等。这样的计算机程序产品可以通过本领域中公知的任何合适的软件安装进程来安装。在另一实施例中，软件指令的至少一部分也可以通过电缆、通信和/或无线连接来下载。

[0129] 进一步地、固件、软件、例程或指令在本文中可以被描述为执行数据处理器的某些动作和/或功能。然而，应当理解，本文中所包含的这种描述仅仅是为了方便，并且这种动作实际上由计算设备，处理器，控制器、或执行固件、软件、例程、指令等的其它设备来产生。

[0130] 还应当理解，流程图、方框图和网络图可以包括不同地布置或者不同地表示的更多或更少的元件。但是还应当理解，某些实现方式可以规定方框和网络图以及图示了实施例的执行的方框和网络图的数目以特定方式来实现。

[0131] 因此，另外的实施例还可以在多种计算机体系结构、物理、虚拟、云计算机和/或其某种组合中实现，并且因此，本文中所描述的数据处理器仅用于说明的目的，而不是作为实施例的限制。

[0132] 尽管已经参照本发明的示例实施例具体示出并且描述了本发明，但是本领域技术人员应当理解，在不背离所附权利要求所涵盖的本发明的范围的情况下，可以在其中进行形式和细节上的各种改变。

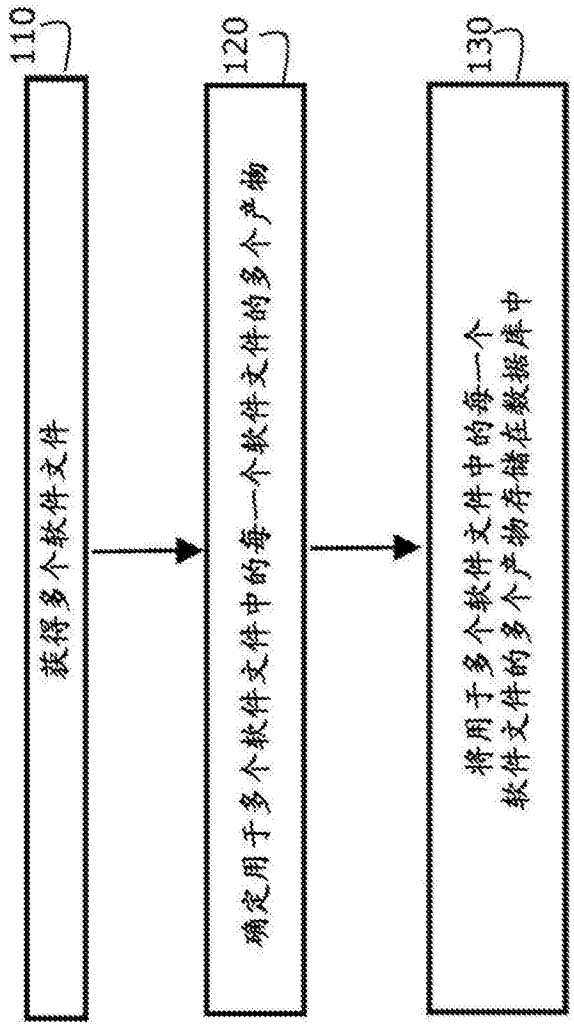

110
获得多个软件文件
120
确定用于多个软件文件中的每一个软件文件的多个产物
130
将用于多个软件文件中的每一个
软件文件的多个产物存储在数据库中

图1

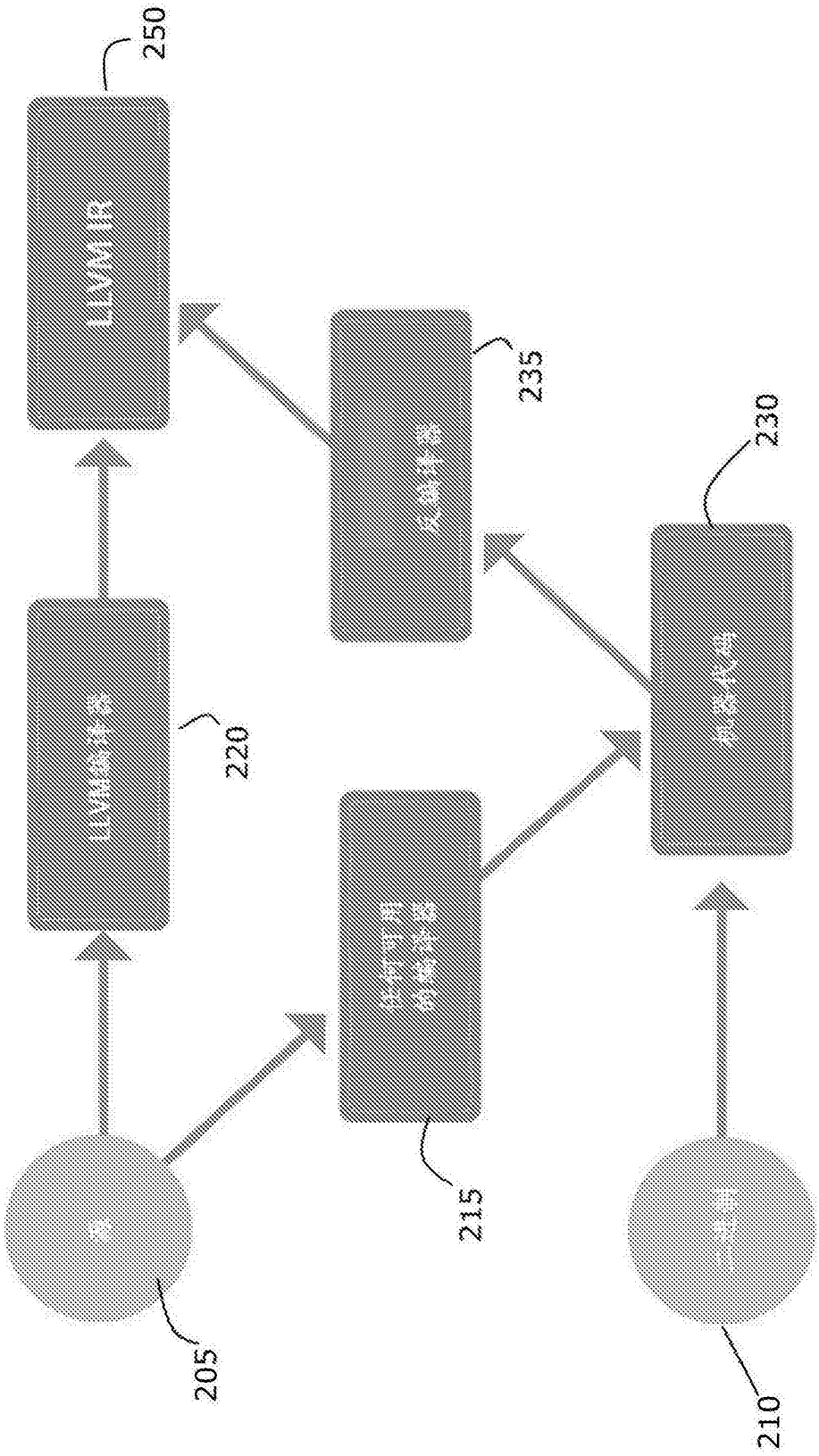


图2

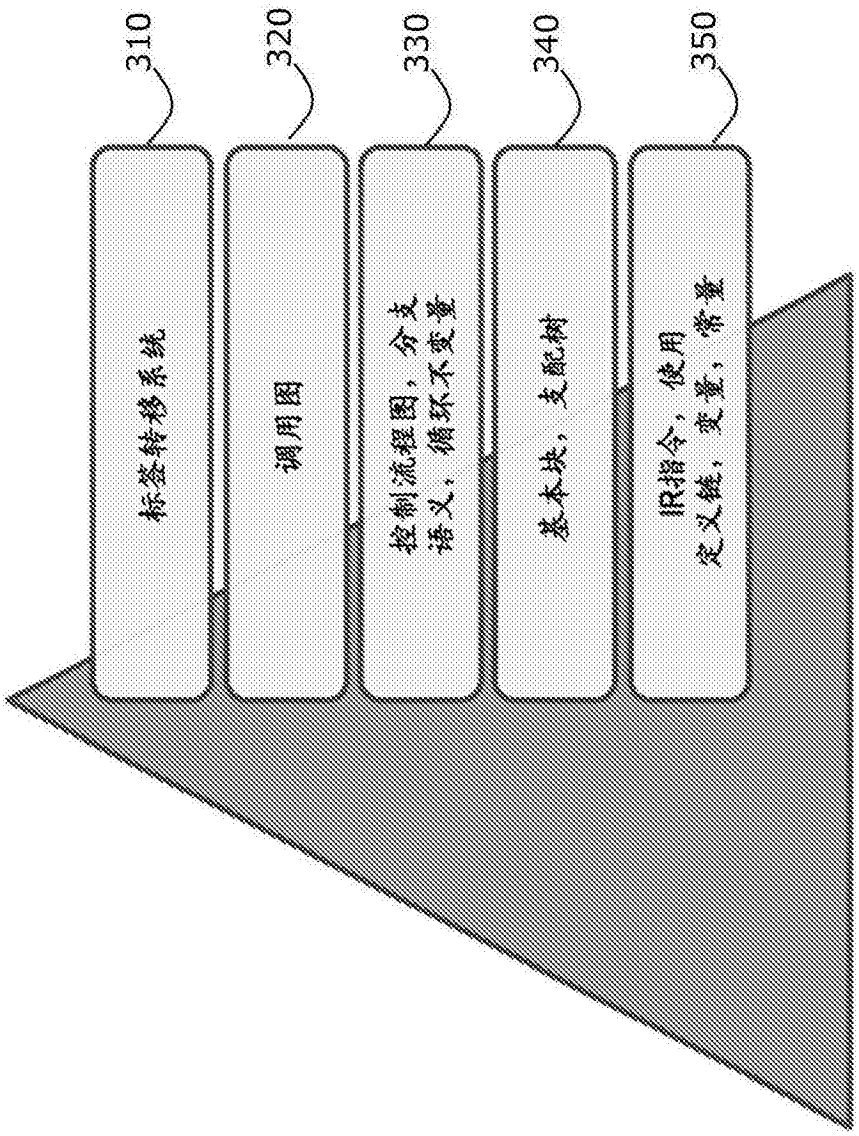

310
320
330
340
350
标签转移系统
调用图
控制流程图，分支
语义，循环不变量
基本块，支配树
IR指令，使用
定义链，变量，常量

图3

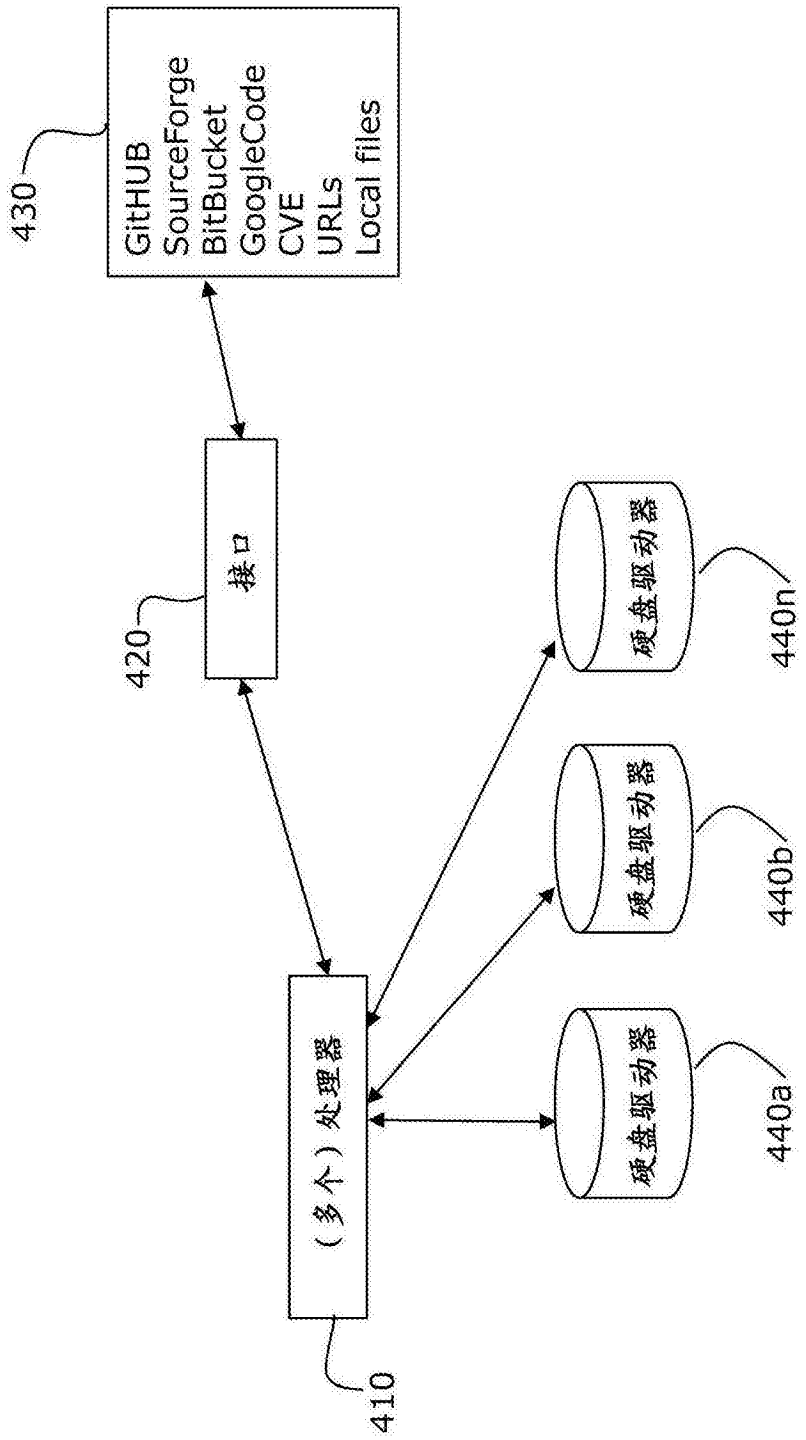

430
GitHUB
SourceForge
BitBucket
GoogleCode
CVE
URLs
Local files
420
接口
410
（多个）处理器
硬盘驱动器
440a
硬盘驱动器
440b
硬盘驱动器
440n

图4

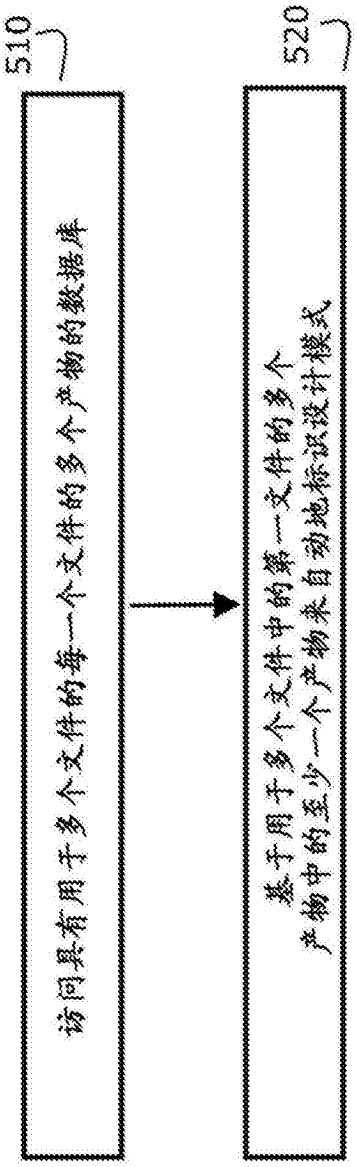

510
访问具有用于多个文件的每一个文件的多个产物的数据库
520
基于用于多个文件中的第一文件的多个产物中的至少一个产物来自动地标识设计模式

图5

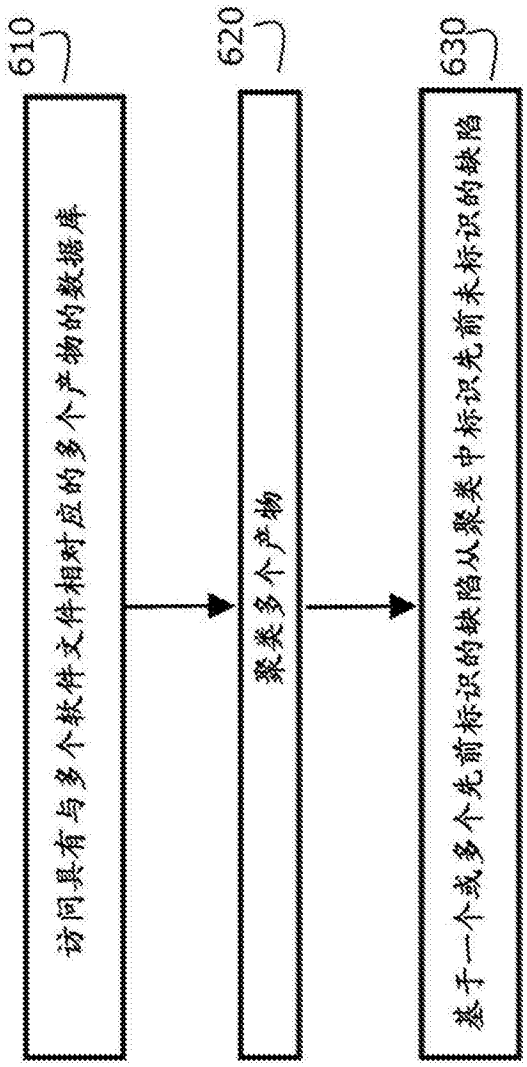

610
访问具有与多个软件文件相对应的多个产物的数据库
620
聚类多个产物
630
基于一个或多个先前标识的缺陷从聚类中标识先前未标识的缺陷

图6

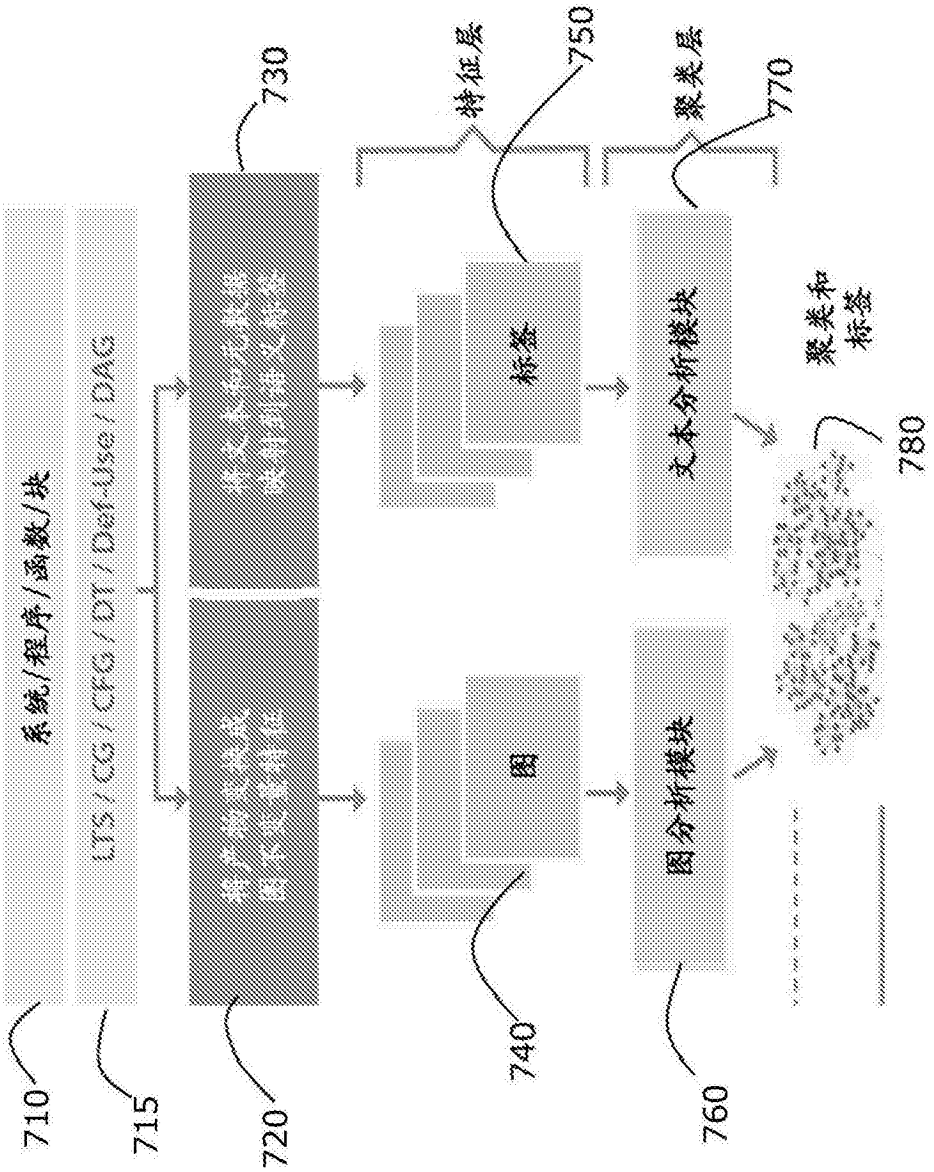

系统/程序/函数/块
LTS / CG / CFG / DT / Def-Use / DAG
730
特征层
750
聚类层
770
标签
文本分析模块
聚类和标签
780
图
图分析模块
710
715
720
740
760

图7

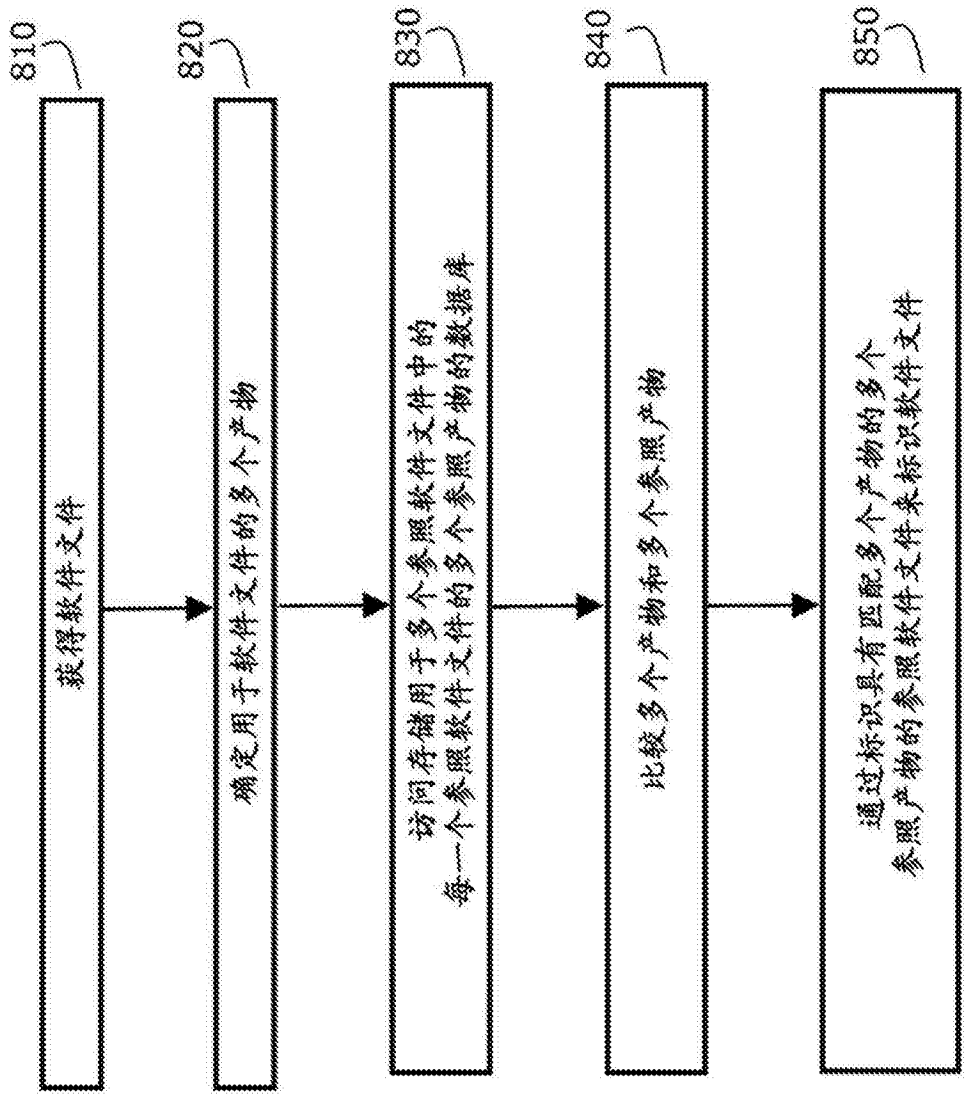

810
获得软件文件
820
确定用于软件文件的多个产物
830
访问存储用于多个参照软件文件中的
每一个参照软件文件的多个参照产物的数据库
840
比较多个产物和多个参照产物
850
通过标识具有匹配多个产物的多个
参照产物的参照软件文件来标识软件文件

图8

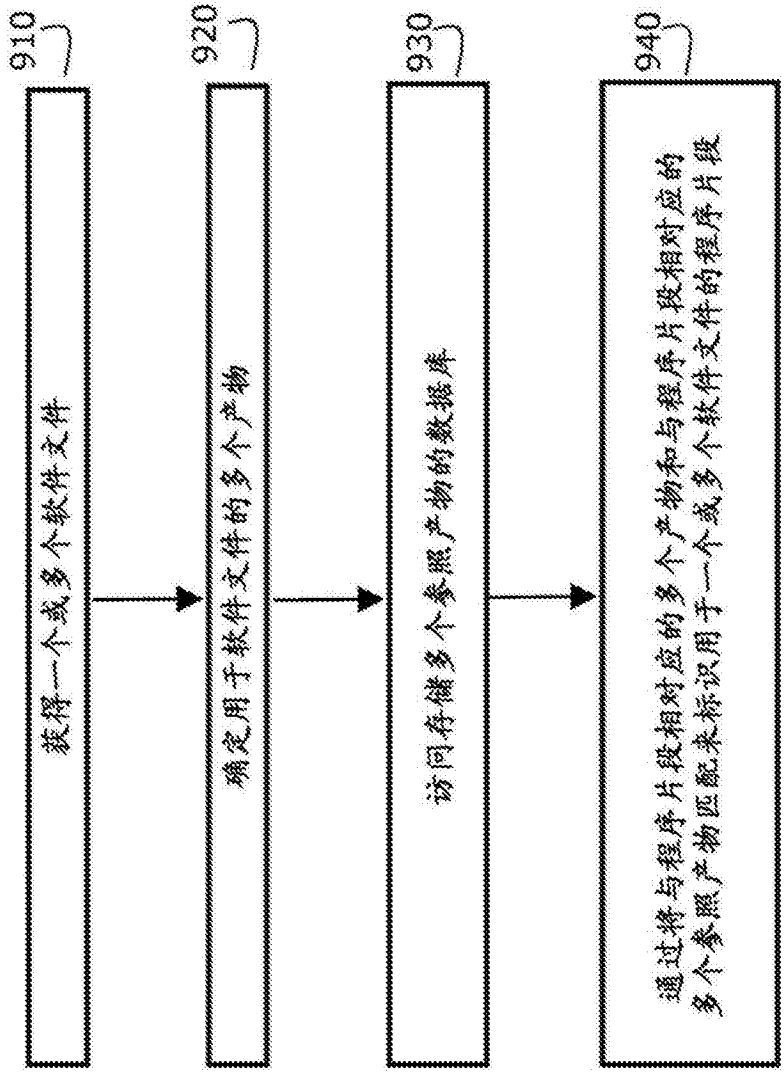

910
获得一个或多个软件文件
920
确定用于软件文件的多个产物
930
访问存储多个参照产物的数据库
940
通过将与程序片段相对应的多个产物和与程序片段相对应的多个参照产物匹配来标识用于一个或多个软件文件的程序片段

图9

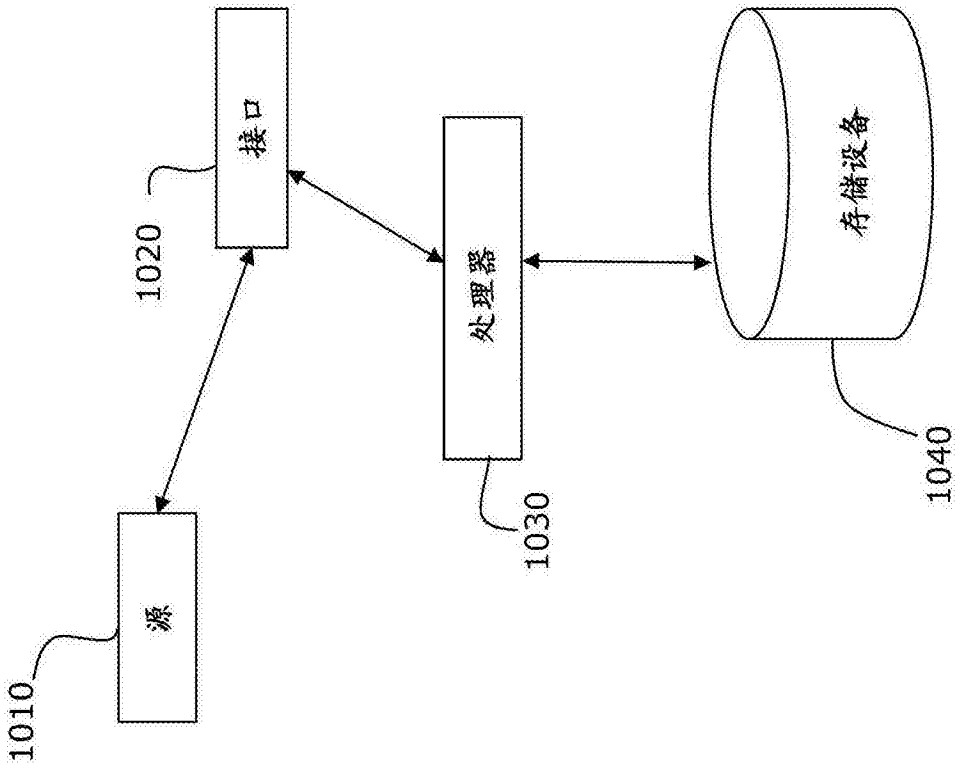


图10