(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 106663003 A
(43)申请公布日 2017.05.10

(21)申请号 201580031458.6
(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/US2015/035138 2015.06.10
(87)PCT国际申请的公布数据
WO2015/191737 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 N·R·什尼德曼
(74)专利代理机构 北京市金杜律师事务所 11256
代理人 王茂华
(51)Int.Cl.
G06F 9/44(2006.01)

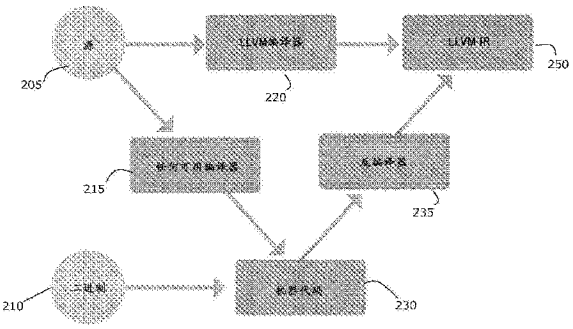
权利要求书4页 说明书17页 附图9页

(54)发明名称

用于软件分析的系统和方法

(57)摘要

提供了用于通过以下方式来标识软件文件、代码中的缺陷和程序片段的系统、方法和计算机程序产品:获取软件文件,确定多个产物,访问存储针对参考软件文件的多个参考产物的数据库,比较产物中的至少一个与存储在数据库中的参考产物中的至少一个,以及通过标识具有对应于多个产物的参考产物的参考软件文件来标识软件文件。某些实施例还可以自动提供文件的更新版本、要应用的补丁或修复的代码块以替换有缺陷的块。示例实施例可以接受各种各样的文件类型,包括源代码和二进制文件,并且可以分析源代码或将文件转换为中间表示(IR)并分析IR。



CN 106663003 A

1.一种用于标识软件的方法,包括:

获取软件文件;

确定针对所述软件文件的多个产物;

访问存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的数据库;

将所述多个产物与所述多个参考产物相比较;以及

通过标识具有与所述多个产物相匹配的所述多个参考产物的所述参考软件文件,来标识所述软件文件。

2.根据权利要求1所述的方法,其中所述多个产物包括以下各项中的一项或多项: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、常数、分支语义和协议。

3.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跟踪和执行跟踪中的一个或多个。

4.根据权利要求1所述的方法,其中所述多个产物包括以下各项中的一项或多项:循环不变量、类型信息、Z语言和标签转移系统表示。

5.根据权利要求1所述的方法,其中所述多个产物包括根据以下各项中的任一项确定的一个或多个产物:内联代码注释、提交历史、文档文件以及常见漏洞和披露源条目。

6.根据权利要求1所述的方法,其中所述多个产物各自是图产物。

7.根据权利要求1所述的方法,其中所述多个产物各自是元数据产物。

8.根据权利要求1所述的方法,其中当所述多个参考产物与所述多个产物之间至少存在模糊匹配时,所述多个参考产物匹配所述多个产物。

9.根据权利要求1所述的方法,其中确定针对所述软件文件的所述多个产物包括:将所述软件文件转换为中间表示,并且根据所述中间表示确定所述多个产物中的至少一个产物。

10.根据权利要求1所述的方法,还包括:通过分析与标识的所述参考软件文件相关联的所述参考产物中的至少一个参考产物,来确定是否存在所述软件文件的较新版本。

11.根据权利要求10所述的方法,还包括自动提供所述软件文件的所述较新版本。

12.根据权利要求1所述的方法,还包括:通过分析与标识的所述参考软件文件相关联的所述参考产物中的至少一个参考产物,来确定是否存在针对所述软件文件的补丁。

13.根据权利要求12所述的方法,还包括向所述软件文件自动应用所述补丁。

14.根据权利要求12所述的方法,还包括:分析所述补丁以确定与所述软件文件中的缺陷的修复对应的所述补丁的修复部分,并且向所述软件文件仅应用所述补丁的所述修复部分。

15.根据权利要求14所述的方法,其中分析所述补丁包括:将所述补丁转换为中间表示,并且根据所述中间表示确定至少一个补丁产物。

16.根据权利要求1所述的方法,还包括:通过分析与标识的所述参考软件文件相关联的所述参考产物中的至少一个参考产物以及与所述软件文件相关联的所述产物中的至少一个产物,来确定所述软件文件中是否存在缺陷。

17.根据权利要求16所述的方法,还包括自动修复所述软件文件中的所述缺陷。

18.根据权利要求17所述的方法,其中自动修复所述缺陷包括使用源代码修复块替换源代码块。

19.根据权利要求17所述的方法，其中自动修复所述缺陷包括使用二进制代码修复块替换二进制代码块。

20.根据权利要求17所述的方法，其中自动修复所述缺陷包括使用中间表示修复块替换所述软件文件中的中间表示块。

21.一种方法，包括：

获取一个或多个软件文件；

确定针对所述一个或多个软件文件的多个产物；

访问存储多个参考产物的数据库；以及

通过将与针对所述一个或多个软件文件的程序片段对应的所述多个产物和对应于所述程序片段的所述多个参考产物相匹配，来标识所述程序片段。

22.根据权利要求21所述的方法，其中所述程序片段已经在所述数据库中被标识为与缺陷对应。

23.根据权利要求21所述的方法，其中所述程序片段与所述一个或多个软件文件中的缺陷对应。

24.根据权利要求21所述的方法，其中所述程序片段与从下列组中选择的缺陷对应，所述组由下列各项组成：故障、安全漏洞和协议缺点。

25.根据权利要求23所述的方法，还包括自动修复所述一个或多个软件文件中的所述缺陷。

26.根据权利要求25所述的方法，其中自动修复所述缺陷包括提供修复程序片段以替换缺陷程序片段。

27.根据权利要求23所述的方法，还包括向用户提供一个或多个修复选项以修复所述缺陷。

28.根据权利要求27所述的方法，还包括对被提供给所述用户的所述一个或多个修复选项排序。

29.根据权利要求28所述的方法，其中所述一个或多个修复选项的所述排序基于由所述用户选择的一个或多个先前修复选项。

30.根据权利要求28所述的方法，其中所述一个或多个修复选项的所述排序基于针对所述修复选项中的每个修复选项的成功的可能性。

31.根据权利要求21所述的方法，其中所述程序片段已经在所述数据库中被标识为与特征对应。

32.根据权利要求31所述的方法，还包括使用特征增强来自动加强所述特征。

33.根据权利要求21所述的方法，其中所述多个产物包括图产物。

34.根据权利要求21所述的方法，其中所述多个产物包括开发产物。

35.根据权利要求21所述的方法，其中所述多个产物各自是元数据产物。

36.根据权利要求21所述的方法，其中确定针对所述一个或多个软件文件的所述多个产物包括：将所述一个或多个软件文件转换为中间表示，并且根据所述中间表示确定所述多个产物中的至少一个产物。

37.根据权利要求21所述的方法，其中所述一个或多个软件文件各自是源代码格式。

38.根据权利要求21所述的方法，其中所述一个或多个软件文件各自是二进制代码格

式。

39.根据权利要求21所述的方法，其中所述一个或多个软件文件是在软件项目内的文件。

40.一种用于标识软件的系统，包括：

接口，能够与具有软件文件的源通信；

存储设备，存储针对多个参考软件文件中的每个参考软件文件的多个参考产物；以及

处理器，通信地耦合至所述接口和所述存储设备，并且被配置为：

引起所述软件文件被获取；

确定针对所述软件文件的多个产物；

访问所述存储设备中的所述多个参考产物；

将所述多个产物与所述多个参考产物相比较；以及

通过标识具有与所述多个产物相匹配的所述多个参考产物的所述参考软件文件，来标识所述软件文件。

41.根据权利要求40所述的系统，其中确定针对所述软件文件的所述多个产物包括：将所述软件文件转换为中间表示，并且根据所述中间表示确定所述多个产物中的至少一个产物。

42.根据权利要求40所述的系统，还包括所述处理器还被配置为通过分析与标识的所述参考软件文件相关联的所述参考产物中的至少一个参考产物，来确定是否存在针对所述软件文件的补丁。

43.根据权利要求40所述的系统，还包括所述处理器还被配置为向所述软件文件自动应用所述补丁。

44.根据权利要求42所述的系统，还包括所述处理器还被配置为分析所述补丁以确定与所述软件文件中的缺陷的修复对应的所述补丁的修复部分，并且向所述软件文件仅应用所述补丁的所述修复部分。

45.一种系统，包括：

接口，能够与具有一个或多个软件文件的源通信；

存储设备，存储多个参考产物；以及

处理器，通信地耦合至所述接口和所述存储设备，并且被配置为：

引起一个或多个软件文件被获取；

确定针对所述一个或多个软件文件的多个产物；

访问存储多个参考产物的数据库；以及

通过将与针对所述一个或多个软件文件的程序片段对应的所述多个产物和对应于所述程序片段的所述多个参考产物相匹配，来标识所述程序片段。

46.根据权利要求45所述的系统，其中所述程序片段已经在所述数据库中被标识为与缺陷对应。

47.根据权利要求45所述的系统，其中所述程序片段与从下列组中选择的缺陷对应，所述组由下列各项组成：故障、安全漏洞和协议缺点。

48.根据权利要求45所述的系统，还包括所述处理器还被配置为自动修复所述一个或多个软件文件中的所述缺陷。

49.一种非暂态计算机可读介质，所述非暂态计算机可读介质上存储有可执行程序，其中所述程序指示处理设备执行以下步骤：

获取软件文件；

确定针对所述软件文件的多个产物；

访问存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的数据库；

将所述多个产物与所述多个参考产物相比较；以及

通过标识具有与所述多个产物相匹配的所述多个参考产物的所述参考软件文件，来标识所述软件文件。

50.一种非暂态计算机可读介质，所述非暂态计算机可读介质上存储有可执行程序，其中所述程序指示处理设备执行以下步骤：

获取一个或多个软件文件；

确定针对所述一个或多个软件文件的多个产物；

访问存储多个参考产物的数据库；以及

通过将与针对所述一个或多个软件文件的程序片段对应的所述多个产物和对应于所述程序片段的所述多个参考产物相匹配，来标识所述程序片段。

用于软件分析的系统和方法

[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第62/012,127号的权益。以上申请的全部教示通过引用合并于此。

[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。软件供应商随着时间计划、实现、记录、测试、部署和维护计算机程序。初始计划、实现、记录、测试和部署通常是不完整的，并且总是缺少所需的功能或包含缺陷。很多供应商有生命周期维护计划以通过随着软件成熟推出迭代故障修复、安全补丁和功能增强来解决这些缺陷。

[0006] 世界上部署了数十亿行的大量的软件代码，并且维护和故障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的(即，响应于故障报告、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，例如故障(代码中的错误)、安全漏洞和协议缺点。本发明的示例实施例提供了可以利用大量软件文件的系统和方法，包括公开可用的或专有的软件文件。

[0008] 某些示例实施例可以自动标识和提供针对软件文件的最新版本或补丁。另外的实施例可以自动定位已知存在于某些软件文件中的设计模式、例如软件缺陷(例如，故障、安全漏洞、协议缺点)并提供修复。其他实施例可以通过先前前不知道文件包含缺陷的软件文件中定位已知缺陷来利用已知缺陷。另外的实施例可以自动定位设计模式，例如标识源代码或二进制代码的部分，以标识文件、程序、函数或代码块。

[0009] 当标识出软件缺陷时，对于一些实施例，可以使用对应的软件修复模式生成修复规范。例如，该修复规范可以用于以源或二进制(也称为机器语言)补丁的形式来合成适当的软件修复。某些示例实施例可以支持对二进制代码和源代码二者执行自动软件维护，例如缺陷标识和修复，从而实现针对传统系统的广泛的自动化软件维护。

[0010] 根据本发明的一个实施例，一种用于标识软件的方法包括获取软件文件，确定针对软件文件的多个产物，访问存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的数据库，将多个产物与多个参考产物进行比较，以及通过标识具有与多个产物匹配的多个参考产物的参考软件文件来标识软件文件。

[0011] 根据另外的实施例，针对软件文件的多个产物可以包括调用图、控制流图、使用定义链、定义使用链、支配树、基本块、变量、常数、分支语义和协议中的一个或多个。对于其他另外的实施例，多个产物可以包括系统调用跟踪和执行跟踪中的一个或多个。对于另一示

例实施例，多个产物可以包括循环不变量、类型信息、Z语言和标签转移系统表示中的一个或多个。对于某些示例实施例，多个产物可以包括根据内联代码注释、提交历史、文档文件和常见漏洞和披露源条目(entry)中的任一个确定的一个或多个产物。对于一些示例实施例，多个产物每个是图产物或开发产物。对于另外的实施例，多个产物每个是静态产物、动态产物、导出的产物或元数据产物。对于某些实施例，当在多个参考产物与多个产物之间至少存在模糊匹配时，多个参考产物匹配多个产物。

[0012] 根据另外的实施例，该方法还可以通过分析存储在数据库中与所标识的参考软件文件相关联的参考产物中的至少一个参考产物来确定是否存在软件文件的较新版本。对于一些实施例，该方法还可以自动提供软件文件的较新版本。

[0013] 根据其他实施例，该方法还可以包括通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考产物来确定是否存在针对软件文件的补丁。某些实施例还可以向软件文件自动应用补丁。其他实施例还可以分析补丁以确定与软件文件中的缺陷的修复对应的补丁的修复部分，并且仅将补丁的修复部分应用于软件文件。对于某些实施例，分析补丁和软件文件包括将补丁转换为中间表示，并且对于某些实施例将软件文件也转换为中间表示，并且根据中间表示确定产物中的至少一个产物。

[0014] 本发明的某些实施例可以通过将软件文件转换为中间表示并且根据中间表示确定多个产物中的至少一个产物来确定针对软件文件的多个产物。另外的实施例还可以在例如虚拟机的仪器化环境中运行软件文件以确定产物。某些实施例还可以通过从软件文件提取字符串来确定产物中的一些产物，包括当软件文件是源代码格式或二进制代码格式时。

[0015] 示例方法的另外的实施例可以通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考产物以及与对于某些实施例的软件文件相关联的产物中的至少一个产物来确定软件文件中是否存在缺陷。另外的实施例可以自动修复软件文件中的缺陷。对于这些实施例中的某些实施例，自动修复缺陷包括用源代码修复块替换源代码块。对于这些实施例中的某些实施例，自动修复缺陷包括用二进制代码修复块替换二进制代码块。对于这些实施例中的某些实施例，自动修复缺陷包括用中间表示修复块替换软件文件的中间表示块。这些块可以是连续的，但不一定是连续的，并且可以包括遍布文件的代码。

[0016] 根据本发明的另一实施例，一种用于标识代码的方法包括获取一个或多个软件文件，确定针对软件文件的多个产物，访问存储多个参考产物的数据库，以及通过将与程序片段对应的多个产物和与程序片段对应的多个参考产物相匹配来标识软件文件中的程序片段。匹配也可以基于模糊匹配，在模糊匹配中，接近匹配被认为是匹配。

[0017] 对于一些实施例，确定针对软件文件的多个产物包括将软件文件转换为中间表示格式，并且根据中间表示确定多个产物中的至少一个产物。对于示例方法的一些实施例，软件文件每个是源代码格式。对于其他实施例，软件文件每个是二进制代码格式。对于一些实施例，程序片段与软件文件中的缺陷对应，例如故障、安全漏洞或协议缺点。对于某些示例实施例，多个产物包括图产物和/或开发产物，或者多个产物每个是元数据产物。对于某些示例实施例，一个或多个软件文件可以是软件项目内的文件。

[0018] 对于某些实施例，与程序片段对应的参考产物先前已经在数据库中被标识为与缺陷对应。对于一些实施例，该方法还包括自动修复软件文件中的缺陷，向用户提供一个或多个修复选项以修复缺陷，和/或对一个或多个修复选项排序，包括基于由用户选择的一个或

多个先前修复选项或基于针对每个修复选项的成功的可能性。自动修复缺陷包括在没有来自用户的针对该文件的任何输入的情况下修复缺陷，包括通过参考配置文件、设置或标志(包括可以由用户(例如管理员)先前设置的那些)来确定是否需要或允许自动修复缺陷。

[0019] 对于某些示例实施例，程序片段已经在数据库中被标识为与特征对应。某些实施例还可以利用特征增强来自动加强特征，包括通过应用二进制或源代码补丁。

[0020] 本发明的另外的实施例提供一种用于标识软件的系统，其包括能够与具有软件文件的源通信的接口、存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的存储设备、通信地耦合到接口和存储设备并且被配置为进行以下操作的处理器：获取软件文件，确定针对软件文件的多个产物，访问存储设备中的多个参考产物，将多个产物与多个参考产物相比较，以及通过标识具有与多个产物相匹配的多个参考产物的参考软件文件来标识软件文件。

[0021] 系统的另外的实施例可以使处理器被配置为除了其他以外通过以下操作来确定针对软件文件的多个产物：将软件文件转换为中间表示，并且根据中间表示确定多个产物中的至少一个产物。其他实施例使处理器还被配置为通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考产物来确定是否存在针对软件文件的补丁。某些另外的实施例使处理器还被配置为将补丁自动应用于软件文件。某些其他实施例使处理器还被配置为分析补丁和软件文件以确定与软件文件中的缺陷的修复对应的补丁的修复部分，并且仅将补丁的修复部分应用于软件文件。

[0022] 本发明的另外的实施例提供了一种用于标识代码的系统，其包括能够与具有一个或多个软件文件的源通信的接口、用于存储多个参考产物的存储设备、以及通信地耦合到接口和存储设备并且被配置为进行以下操作的处理器：引起一个或多个软件文件被获取，确定针对一个或多个软件文件的多个产物，访问存储多个参考产物的数据库，以及通过将与程序片段对应的多个产物和与程序片段对应的多个参考产物相匹配来标识针对一个或多个软件文件的程序片段。对于某些示例实施例，程序片段已经在数据库中被标识为与缺陷对应。这种缺陷的示例包括故障、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相关。另外的实施例也可以使处理器被配置为自动修复一个或多个软件文件中的缺陷。

[0023] 根据本发明的另一实施例，提供了一种其上存储有可执行程序的非暂态计算机可读介质，其中程序指令处理设备执行以下步骤：获取软件文件，确定针对软件文件的多个产物，访问存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的数据库，将多个产物与多个参考产物进行比较，以及通过标识具有与多个产物相匹配的多个参考产物的参考软件文件来标识软件文件。

附图说明

[0024] 根据如附图中所示的本发明的示例实施例的以下更具体的描述，上述内容将是显而易见的，其中在不同的视图中相同的附图标记表示相同的部分。附图不一定是按比例的，重点在于示出本发明的实施例。

[0025] 图1是示出用于提供针对软件文件的语料库的方法的示例实施例的流程图。

[0026] 图2是示出根据本发明的实施例的用于从针对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。

[0027] 图3是示出根据本发明的实施例的用于软件文件的产物之间的层级关系的框图。

[0028] 图4是示出用于提供针对软件文件的产物的语料库的系统的示例实施例的框图。

[0029] 图5是示出用于标识设计模式的方法的示例实施例的框图。

[0030] 图6是示出用于标识缺陷的方法的示例实施例的流程图。

[0031] 图7是示出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。

[0032] 图8是示出用于使用语料库来标识软件文件的方法的示例实施例的流程图。

[0033] 图9是示出用于标识程序片段的方法的示例实施例的流程图。

[0034] 图10是示出根据本发明的实施例的使用语料库的系统的框图。

具体实施方式

[0035] 以下是本发明的示例实施例的描述。本文引用的任何专利或出版物的全部教导通过引用并入本文档中。

[0036] 根据本公开的示例实施例的软件分析允许利用来自现有软件文件的知识,包括来自公共可用来源或者是专有软件的文件。然后,该知识可以应用于其他软件文件,包括以修复缺陷,标识漏洞,标识协议缺点或建议代码改进。

[0037] 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、维护或以其他方式提供针对知识数据库的软件文件的语料库和关于软件文件的相关产物。根据本发明的方面,该语料库可以用于各种目的,包括自动标识软件文件的较新版本、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、以及在以前未知包含这些错误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。

[0038] 图1是示出根据本发明的实施例的针对语料库的输入软件文件的示例处理的流程图。第一图示步骤是获取多个软件文件110。这些软件文件可以是源代码格式,其通常是纯文本、或者以二进制代码格式或某种其他格式。此外,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。对于某些另外的示例实施例,还可以获取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。

[0039] 所获取的软件文件不仅包括源代码或二进制文件,还可以包括与这些文件或对应的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、make文件、库、文档文件、提交日志、修订历史、bugzilla条目、常见漏洞和披露(CVE)条目和其他非结构化文本。

[0040] 可以从各种源获取软件文件。例如,可以经由因特网通过网络接口从例如GitHUB、SourceForge、BitBucket、GoogleCode或(例如由MITRE公司维护的)常见漏洞和披露系统的公共可用软件存储库获取软件文件。通常,这些存储库包含文件和对文件所进行的改变的历史。此外,例如,可以提供统一资源定位符(URL)以指向可以从其获取文件的站点。软件文件还可以经由接口从专用网络或本地地从本地硬盘驱动器或其他存储设备来获取。接口提供用于通信地耦合到源。

[0041] 本发明的示例实施例可以从源获取一些、大多数或所有可用文件。此外,一些示例实施例还自动获取文件,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、或从源可获得的所有文件。一些实施例爬过针对整个存储库的每个修订以获取所有可用软件文件。某些示例实施例为语料库中的每个软件项目获取整个源控制存储库,以支持自动获取针对项目的所有相关联的文件,包括获取每个软件文件修订。针对存储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返回检查源以辨别源是否已经改变或更新,并且如果是,则可以仅从源获取改变或更新,或者也可以再次获取所有软件文件。很多源具有用于确定对源的改变的方法,例如示例实施例可以用于从源获取更新的日期添加或日期改变字段。

[0042] 本发明的某些示例实施例还可以单独获取库软件文件,这些库软件文件可以由从存储库获取的源代码文件使用以在存储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取从任何公共源合理地可获取的或从软件供应商获取的任何库软件文件以包括在语料库中。另外,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,使得能够获取这些库。某些实施例刮除(scrape)针对每个项目的软件文件以标识由项目使用的库,使得能够根据需要来获取并且也安装这些库。

[0043] 根据本发明的示例方法中的下一步骤是确定针对多个软件文件120中的每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、导出的产物和元数据产物。

[0044] 示例方法的最后一步是将针对多个软件文件中的每个软件文件的多个产物存储在数据库130中。多个产物以如下方式存储:该方式使得这些产物能够被标识为与能够根据其来确定这些产物的特定软件文件对应。可以以众所周知的各种方式中的任何一种来完成该标识,例如由数据库模式表示的数据库中的字段、指针、存储位置或者例如文件名等任何其他标识符。可以类似地跟踪属于同一项目或构建的文件,使得能够维持关系。

[0045] 对于不同的实施例,数据库可以采取不同的形式,例如图数据库、关系数据库或平面文件。一个优选实施例使用OrientDB,其是由Orient Technologies领导的OrientDB开源项目提供的分布式图数据库。另一优选实施例使用Titan(其是被优化用于存储和查询分布在多机器集群上的图的可扩展图数据库)以及Apache Cassandra存储后端。某些示例实施例还可以使用SciDB,其是来自范例4的也存储和操作图产物的数组数据库。

[0046] 静态产物、动态产物、导出的产物和元数据产物通常可以根据源代码文件、二进制文件或其他产物来确定。下面提供这些类型的产物的示例。示例实施例可以确定针对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。某些实施例没有确定这些类型的产物中的每个产物或针对特定类型的产物中的每个产物,而是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/或没有确定任何具体的类型。

[0047] 静态产物

[0048] 针对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、分支语义和协议。

[0049] 调用图(CG)是被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是定向的并且示出函数是否可以调用另一函数。

[0050] 控制流图(CFG)是在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，节点之间的边是定向的，并且示出流中的潜在路径。

[0051] 用户定义链(UD)和定义用户链(DU)是在基本代码块中执行的输入(使用)、输出(定义)和操作的有向无环图。例如，UD链是变量的使用、以及可以达到该使用而没有中间的重新定义的该变量的所有定义。DU链是变量的定义以及可以从该定义达到而没有中间的重新定义的所有使用。这些链使得能够关于所接受的输入类型、所生成的输出类型以及在基本代码块内执行的操作实现基本代码块的语义分析。

[0052] 支配树(DT)是表示CFG中的哪些节点主导其他节点(在其他节点的路径中)的矩阵。例如，如果从入口节点到第二节点的每个路径必须通过第一节点，则第一节点支配第二节点。DT用Pre(从入口前进)和Post(从出口后退)形式来表示。当路径改变到CFG中的特定节点时，DT突出显示。

[0053] 基本块是CFG的每个节点内的指令和操作数。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。

[0054] 变量对于任何函数参数、局部变量或全局变量是针对信息及其类型(表示其可以存储的信息的类型)的存储单位，并且包括默认值(如果有的话)。它们可以提供关于程序的初始状态和基本约束，并且示出类型或初始值的改变，这可以影响程序行为。

[0055] 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。它们可以示出类型或初始值的改变，这可以影响程序行为。

[0056] 分支语义是if语句和循环内的布尔估计。分支控制执行它们的基本块的条件。

[0057] 协议是协议、库、系统调用和由程序使用的其他已知函数的名称和引用。

[0058] 本发明的示例实施例可以根据例如由公共可用的LLVM(以前的低级虚拟机)编译器基础设施项目提供的软件源代码文件的中间表示(IR)自动确定静态产物。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)，例如ARM、X86、X64、MIPS和PPC。可以使用针对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。至少针对Ada、C/C++、D、Erlang、Haskell、Java、Lua、Objective C/C++、PHP、Pure、Python和Ruby的前端是公开可用的。此外，可以容易地编程针对另外的语言的前端。LLVM还具有可用的优化器以及可以将LLVM IR转换为针对各种不同ISA的机器语言的后端。另外的示例实施例可以根据源代码文件确定静态产物。

[0059] 图2是示出根据本发明的实施例的可以使用的针对语料库的输入软件文件的另外的示例处理的流程图。除了其他以外，示例实施例可以获取源代码205和二进制代码210软件文件二者。当LLVM编译器220可用于源代码文件205的语言时，可以使用针对该语言的LLVM编译器220将源代码转变成LLVM IR 250。对于没有可用LLVM编译器的编译语言，可以首先使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，使用例如Fracture等反编译器235来反编译二进制文件230，Fracture是由Draper Laboratory提供的公开可用的开源反编译器。反编译器235将机器代码230转变成LLVM IR 250。对于以二进制形式210获取的文件(其是机器代码230)，使用反编译器235对它们进行反编译以获取LLVM IR 250。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。

[0060] 本发明的示例实施例可以自动获取针对每个源代码软件文件的IR。例如，示例实施例可以在存储库中自动搜索针对标准构建文件(例如autocomf、cmake、automake或make

文件或供应商指令)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前端调用来自动地选择性地尝试使用这样的文件来构建项目。针对构建文件的选择过程可以遍历每个文件以确定哪个文件存在并提供完成的构建或部分完成的构建。

[0061] 另外的示例实施例可以在从存储库自动获取文件、将文件转换为LLVM IR和/或确定针对文件的产物时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向从属机器推送项目和构建以进行处理。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转变为LLVM IR和/或确定产物并且提供结果用于存储在语料库中。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源软件框架。从源存储库获取文件也可以分布在一组机器中。

[0062] 根据示例实施例，软件文件和LLVM IR也可以被存储在语料库中，包括被存储在分布式存储中。示例实施例还可以确定软件文件或LLVM IR代码已经被存储在数据库中并且选择不再次存储文件。指针、图数据库中的边或其他引用标识符可以用于将文件与特定项目、目录或其他文件集合相关联。

[0063] 动态产物

[0064] 动态产物表示程序行为，并且通过在例如虚拟机、仿真器(例如，快速仿真器(“QEMU”))或管理程序等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。

[0065] 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请求服务，内核管理输入/输出请求。库调用是对软件库的调用，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。

[0066] 执行跟踪是包括指令字节、堆栈帧、存储器使用(例如，驻留/工作集大小)、用户/内核时间和其他运行时间信息的每指令跟踪。

[0067] 本发明的示例实施例可以产生虚拟环境(包括针对各种操作系统的)，并且可以运行和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，可以使用例如Valgrind或Daikon等公共可用程序来提供关于程序的运行时间信息以用作产物。Valgrind是除了其他以外用于调试存储器、检测存储器泄漏和分析。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某些点处成立的条件。

[0068] 其他实施例可以使用公开可用的另外的诊断和调试程序或实用程序，例如strace和dtrace。Strace用于监视进程与内核之间的交互，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)。

[0069] 另外的实施例可以通过KLEE引擎运行LLVM IR。KLEE是符号虚拟机，其是公开可用的开源代码。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。符号执行除了其它以外涉及分析代码以确定什么输入引起代码的每个部分执行。使用KLEE在发现功能正确性错误和行为不一致性方面非常有效，因此使得本发明的示例实施例能够快速地标识类似代码中的差异(例如，跨修订)。

[0070] 导出的产物

[0071] 导出的产物代表复杂的高级程序行为并且提取表征这些行为的属性和事实。导出

的产物包括程序特性、循环不变量、扩展类型信息、Z语言和标签转移系统表示。

[0072] 程序特性是关于根据执行跟踪导出的程序的事实。这些事实包括最小、最大和平均存储器大小;执行时间;和堆栈深度。

[0073] 循环不变量是在循环的所有迭代(或选择的迭代组)上被维持的属性。循环不变量可以被映射到分支语义以揭示类似的行为。

[0074] 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、与其他变量的关系以及可以被抽象的其他特征。类型约束可以揭示关于代码的行为和功能。

[0075] Z语言基于Zermelo-Fraenkel集合理论。其提供了类型代数符号,以实现基本块与整个函数之间的比较度量,而忽略结构、顺序和类型。

[0076] 标签转移系统(LTS)表示是表示根据程序抽象的高级状态的图系统。图的节点是状态,并且边用转移中的相关联的动作来标记。

[0077] 对于某些示例实施例,可以根据其他产物,根据源代码文件(包括使用上面描述的针对动态产物的程序)并且根据LLVM IR来确定导出的产物。

[0078] 元数据产物

[0079] 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,例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相关的产物。开发产物可以包括内联代码注释、提交历史、bugzilla条目、CVE条目、构建信息、配置脚本和文档文件,例如README.*TODO.*。

[0080] 示例实施例可以使用Doxygen,其是公开可用的文档生成器。Doxygen可以根据特别注释的源代码文件(即内联代码文档)为程序员和/或最终用户生成软件文档。

[0081] 另外的实施例可以使用解析器,例如用于语言识别的另一种工具(ANTLR)4生成的解析器,以产生抽象语法树(AST)从而提取高级语言特征,其也可以用作产物。ANTLR4采用针对语言的字符串的语法、产生规则,并且生成可以构建和运行(walk)解析树的解析器。得到的解析器发出各种类型、函数定义/调用以及与程序结构相关的其他数据。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,从针对每个范例)和结构化注释(例如,正式的前/后条件语句)。示例实施例可以将该提取的数据映射到其在LLVM IR中的被引用位置,因为文件名、行和列号信息存在于解析器和LLVM IR两者中。

[0082] 本发明的示例实施例可以通过从源软件文件提取字符串(例如,内联注释)来自动确定一个或多个元数据产物。其他实施例自动确定来自文件系统或源控制系统的元数据产物。

[0083] 层级的产物间关系

[0084] 图3是示出根据本发明的实施例的针对软件文件的产物之间的层级关系的框图。示例实施例可以维护和利用这些层级的产物间关系。此外,不同的实施例可以使用不同的模式和不同的层级关系。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可以映射到函数和特定的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可以利用CFG产物330映射到特定函数,CFG产物330的边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。在这些产物下面是变量、常数、UD/DU链和IR指令350。图3清楚地示出了产物可以被映射到层级的不同级别,从描

述动态信息的范围向下直到单独的IR指令的LTS节点。这些层级关系可以由示例实施例用于各种用途，包括更有效地搜索匹配产物，例如通过首先比较更靠近层级顶部的产物(与更接近底部的产物相比)以便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，这取决于较高级产物是否是匹配。另外的实施例还可以在针对缺陷或特征增强定位或建议修复代码时利用层级关系，包括通过在层级中进行到更高来定位针对具有匹配的更高级产物的缺陷的修复代码。

[0085] 图4是示出用于提供针对软件文件的产物的语料库的系统的示例实施例的框图。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，例如本地硬盘驱动器或盘。在其他实施例中，接口420可以是用于通过公共或专用网络获取文件的网络接口420。这些软件文件的公共源430的示例包括GitHUB、SourceForge、BitBucket、GoogleCode或常见漏洞和披露系统。专用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存储库中。该示例系统还具有耦合到接口420以从源430获取多个软件文件的一个或多个处理器410。处理器410还可以用于确定针对多个软件文件中的每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、导出的产物和/或元数据产物。对于另外的实施例，处理器410还可以被配置为将每个软件文件转换为中间表示并且根据中间表示确定产物。

[0086] 示例系统还具有一个或多个存储设备440a到440n，其用于存储针对每个软件文件的产物，并且耦合到处理器410。这些存储设备440a到440n可以是硬盘驱动器、硬盘驱动器阵列、其他类型的存储设备和分布式存储，例如通过在Hadoop文件系统(HDFS)上使用Titan和Cassandra提供的。同样，示例系统可以具有一个处理器410或者采用分布式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在接口420与存储设备440a到440n之间的直接通信耦合。

[0087] 图5是示出用于定位设计模式的方法的示例实施例的框图。设计模式的示例包括故障、修复、漏洞、安全补丁、协议、协议扩展、功能和功能增强。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。

[0088] 示例方法提供访问具有与多个软件文件对应的多个产物的数据库510。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、或通过因特网或云可用。一旦已经访问了数据库，则该方法可以基于针对多个文件中的第一文件的多个产物中的至少一个产物自动标识设计模式520。对于某些示例实施例，多个产物中的每个产物可以是静态产物、动态产物、导出的产物或元数据产物。其他实施例可以具有不同类型的产物的混合。此外，文件的格式不受限制，并且例如可以是二进制代码格式、源代码格式或中间表示(IR)格式。

[0089] 对于某些实施例，可以通过开发产物的关键字搜索或自然语言搜索来标识设计模式。例如，源代码文件的修订中的内联代码注释可以标识被发现和修复的缺陷。评论可以使用例如缺陷、故障、错误、问题、缺点或失灵等词。这些字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，例如以解决缺陷或增强特征。此外，可以将训练和反馈应用于搜索以改进搜索结果。

[0090] 另外的示例实施例可以从CVE源搜索开发产物，其标识文本中的常见漏洞和错误，

并且可以描述缺陷和可用修复(如果有的话)。该文本可作为产物被获取并且被存储在数据库中。某些源对缺陷编码,使得代码可以用作关键字来定位哪个文件包含缺陷。另外,可以在软件文件的标识中考虑和加权产物的源。例如,与没有溯源或内联注释的存储库相比,CVE源在标识缺陷中可能更可靠。其他实施例可以使用例如文件名和修订号的元数据产物来至少初步标识软件文件,并且基于匹配另外的产物(例如CG或CFG)来确认标识。

[0091] 本发明的某些实施例执行示例方法,并且尝试标识针对一些、大多数或所有源代码和LLVM IR文件的设计模式。另外,每当将文件被添加到语料库时,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。

[0092] 某些实施例还找到与已经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,开发产物可以规定源代码中哪里存在缺陷以及补丁中哪里存在修复。此外,可以分析源代码或LLVM IR,并且将其与具有缺陷的文件和文件的较新修复版本进行比较,以隔离差异并辨别缺陷和修复的位置。对于某些实施例,在开发产物中标识的缺陷类型也可以用于缩小针对缺陷位置的代码的搜索。另外的实施例还可以标识设计模式,例如使用标签,并且将标识符存储在针对文件的数据库中。这使得数据库能够容易地搜索某些缺陷或某些类型的缺陷。这样的标签的示例包括从针对软件文件的开发产物或从源代码获取的字符串。该相同的方法可以应用于标识特征和特征增强并且对它们进行标记。

[0093] 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,设计模式可以涉及文件之间的交互,例如接口。示例实施例可以通过使标识基于针对多个软件文件(例如都属于软件项目的第一和第二文件)的产物来自动标识设计模式。例如,表示设计模式(例如,接口不匹配错误)的预先标识的模式可以被存储在数据库中或其他地方,这些地方使得能够将来自第一和第二文件的产物用于标识针对这些文件存在接口错误。针对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。

[0094] 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(例如故障、错误或缺陷)、以及关于修复的字符串以及在代码中可以找到字符串的位置存在于开发产物中。这些开发产物还可以具有表示特征或特征增强的字符串。

[0095] 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在先前标识,或者可以以某种其他方式标识。这些预先标识的模式可以对应于缺陷、修复、特征、特征增强或感兴趣的项目或其他重要性。

[0096] 图6是示出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对应的多个软件产物的数据库,例如语料库610。然后,分析产物以从数据量中标识模式。例如,该分析可以包括聚类多个产物620。通过聚类数据,可以找到不被知道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标识先前未标识的缺陷630。

[0097] 本发明的某些示例实施例可以对语料库使用机器学习。机器学习涉及通过以下方式来学习数据的层级结构: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。某些示例实施例可以对语料库使用深度学习。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家族的子集。对于某些实施例,可以使用自动编码器用于聚类。

[0098] 对于某些示例实施例,可以由一组自动编码器处理产物以自动发现未标记图和文

档产物的紧凑表示。图产物包括可以以图形式表示的那些产物，例如CG、CFG、UD链、DU链和DT。然后可以聚类图产物的紧凑表示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，故障、修复、漏洞、安全补丁、协议、协议扩展、特征和特征增强)。

[0099] 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，首先以矩阵形式表示所提取的图产物。很多提取的产物可以表示为邻接矩阵，包括例如CFG、UD链和DU链。可以在软件文件和项目层级结构的每个级别处学习结构特征。

[0100] 图产物中的节点的数目可以广泛地变化；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前k个特征值，以使得深度学习能够执行类似于频谱聚类的处理。其他中间产物包括聚类系数，以提供图中的节点趋向于聚类在一起的程度的度量，例如全局聚类系数、网络平均聚类系数和传递率。另一个中间产物是图的荫度(arboricity)，即图如何密集的度量。具有很多边的图具有高的荫度，具有高荫度的图具有密集的子图。另一中间产物是等周数，即图是否具有瓶颈的数值度量。这些中间产物捕获图的结构的不同方面用于在机器学习方法中使用。

[0101] 机器学习、包括深度学习，例如实施例可以采用使用如下多步骤过程训练的算法：其从简单的自动编码器结构开始，并且迭代地改进该方法以开发SSAE。SSAE还可以被训练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。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的输入和输出，其学习身份函数的近似。自动编码器将输入信号脱水(编码)为一组基本的描述性参数，并且对这些信号进行再水化(解码)以重新创建原始信号。描述性参数可以在训练期间被自动选择以优化所有训练信号的再水化。脱水信号的基本性质提供了用于将信号分组成簇的基础。

[0102] 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均值算法聚类所学习的特征。k均值算法是迭代改进技术，其将特征分成k个簇，这使所得到的簇均值最小化。可以基于提取的主题的数目来选择簇的初始数目k。搜索潜在簇的数目非常有效，为很多不同k中的每个计算新的结果，因为针对k均值聚类的操作度量基于欧几里得距离。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对所得到的簇分类。

[0103] 尽管特征向量是稀疏和紧凑的，但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，例如针对“修复”代码，参数空间中的模式应当出现。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合并到自动编码器中。具体地，当标签被系统学习时，示例实施例可以将该信息合并到自动编码器操作中。

[0104] 示例实施例可以使用数据库管理(例如，连接、过滤器)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原语用于特征提取。SVD还可以用于对用于学习算法的输入数据进行去噪，并且使用更少的维度来近似数据，并且因此执行数据简化。

[0105] 示例实施例可以通过文档产物的无监督语义标签生成(包括通过文本分析)来封

装人随着时间和跨程序对代码状态的理解。文本分析的示例是潜在狄利克雷分配(LDA)。可以使用LDA和主题建模从文档产物中提取语义信息。这些方法是“词袋”技术，这些技术考虑单词或短语的出现，而忽略顺序。例如，表示“科学计算”的袋子可以具有例如“FFT”、“小波”、“sin”和“atan”等种子术语。示例实施例可以使用来自源的提取的文档产物，例如源评论、CG/CFG节点标签，并且通过对术语的出现进行计数来提交消息以填充“袋子”。所得到的固定区间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。提取的主题捕获与所提取的文档产物相关联的语义信息，并且可以用作通过经由自动编码器的图产物的无人监督学习形成的簇的标签(例如，故障/修复、漏洞/补丁)。可以由另外的示例实施例使用的其他形式的文本分析包括自然语言处理、词法分析和预测分析。

[0106] 从文档产物提取的主题标签可以提供标签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的主题、表示顺序软件模式(即，在软件修订之前/之后)的语义共性来在语料库数据库中查询训练数据群体。这些模式可以捕获嵌入软件开发文件(例如提交日志、更改日志和注释)中的改变，这些改变随着时间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(例如故障/修复、漏洞/安全补丁和特征/增强)相关的软件的演变的深入理解。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。

[0107] 图7示出了图示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。可以在软件文件层级结构(包括系统、程序、函数和框710)的每个级别处学习结构特征。可以针对聚类715分析图产物，例如CG，CFG和DT。这些图产物然后可以转换成图不变量特征720。这些图特征740然后可以被提供作为图分析模块760(例如自动编码器)的输入、以及所得到的聚类审查被聚类在一起的类似的设计模式780。可以将文本(例如来自源代码文件或来自开发产物的一个或多个字符串)映射到标签730。这些标签750可以由文本分析模块770来分析，例如通过使用LDA或其他自然语言处理，并且标签可以与从其导出标签的对应发现的簇780相关联。这些模块760、770可以用软件，硬件或其组合来实现。

[0108] 图8示出了图示用于使用语料库标识软件的方法的示例实施例的流程图。该示例实施例获取软件文件810。文件可以经由网络接口从公共或专用源来获取，例如经由因特网、云或私人公司的服务器从公共存储库来获取。某些示例实施例还可以从本地源(例如本地硬盘驱动器、便携式硬盘驱动器或盘)获取软件文件。示例实施例可以从源获取单个文件或多个文件，并且可以例如通过使用脚本语言自动地或者通过用户交互手动地这样做。示例方法然后可以确定针对软件文件的多个产物820，例如本文所描述的任何其他产物。示例方法然后可以访问数据库830，数据库存储针对多个参考软件文件中的每个参考软件文件的多个参考产物。参考产物可以被存储在语料库数据库中。对于某些示例实施例，这些参考文件可以包括先前已经被获取的并且其产物(对于某些实施例连同软件文件一起)已经被存储在数据库中的软件文件。将针对所获取的软件文件已经确定的产物或其多个子集与被存储在数据库中的参考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与多个产物相匹配的多个参考产物的参考软件文件来标识软件文件850。因为比较的产物和参考产物匹配，所以软件文件和参考软件文件被标识为是相同的文件。

[0109] 然后可以比较另外的产物或代码部分以增加做出正确标识的置信水平。置信度可以是固定的或可调整的，并且可以基于各种各样的标准，例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该调整。此外，对于某些实施例，匹配可以包括模糊匹配，例如具有小于100％匹配的百分比的可调节设置，以具有声明的匹配。

[0110] 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，常见产物、例如指令是否与32位还是64位处理器相关联可以被给予权重零或一些其他更小权重。一些产物在转换下可以是或多或少不变的，并且对于某些示例实施例，可以相应地调整针对这些产物的权重。例如，文件名或CG产物可以被认为在建立文件的身份中是高信息性的，而某些产物(例如LTS或DT)例如可以被认为是更少决定性的并且对于某些示例实施例和源被给予更小的权重。另外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，与使得基本块产物和DT产物相匹配相比，使得CFG和CG产物相匹配可以在进行标识时被给予更多的权重。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。在标识过程中评估权重的另外的示例可以包括表示标识阈值，例如以匹配产物的百分比或一些其他度量。另外的实施例可以改变标识阈值，包括基于例如文件的源、文件的类型、时间戳(包括文件的日期)、文件的大小、或者某些产物是否针对该文件不能确定或以其他方式不可用。

[0111] 另外的实施例可以通过将软件文件转换为例如LLVM IR的中间表示并且根据中间表示确定多个产物中的至少一个产物来确定针对软件文件的多个产物中的一些产物。其他实施例可以通过从软件文件中提取字符串(例如源代码文件或文档文件)来确定多个产物中的一些产物。

[0112] 示例实施例还可以包括通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考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，则可以检查数据库以查看软件文件的较新修订是否可用，例如通过检查对应参考文件的修订号或时间戳、或者可以将参考文件标识为另一文件的更老修订的与数据库中的产物和文件相关联的标签。另外的示例实施例还可以自动提供软件文件的较新版本，包括给用户或公共或专用源。

[0113] 某些另外的实施例可以通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考产物来确定是否存在针对软件文件的补丁。例如，示例实施例可以检查与参考软件文件相关联的产物，并且确定针对文件存在补丁，包括尚未应用于软件文件的补丁。另外的实施例可以将补丁自动应用于软件文件或者提示用户他们是否想要应用补丁。

[0114] 某些另外的实施例可以分析补丁以及针对某些实施例还可以分析软件文件(或者参考软件文件，因为它们匹配)，以确定与软件文件中缺陷的修复对应的补丁的修复部分。针对某些实施例，该分析可以在获取软件文件之前或之后发生。另外的实施例可以仅将补丁的修复部分应用于软件文件，包括自动或提示用户他们是否想要应用补丁的修复部分。另外的实施例可以将补丁的修复部分提供给源以在源处应用补丁。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为中间表示，并且根据中间表示确定多个产物中的至少一个产物。类似地，另外的实施例可以分析补丁和软件文件(或者参考软件文件，因为它们被匹配)，以确定与软件文件中的特征的改进或改变对应的补丁的特征增强部分。另外的实施例可以仅将补丁的特征增强部分应用于软件文件，包括自动或提示用户他们是否想要应用补丁的特征增强部分。

[0115] 另外的示例实施例可以通过分析与所标识的参考软件文件相关联的参考产物中的至少一个参考产物来确定软件文件中是否存在缺陷。例如，参考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。另外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自动替换源代码块或者用中间表示的修复块自动替换软件文件中的中间表示块。另外的实施例可以通过用二进制补丁替换二进制文件的部分来修复二进制文件中的缺陷。对于某些实施例，修复的文件可以被发送到软件文件的源。另外的实施例可以提供要被提供给软件文件的源修复代码，以便在源处修复文件。

[0116] 图9是示出用于标识代码的方法的示例实施例的流程图。该示例方法可以获取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，则某些实施例可以替代地获取产物，而不是确定产物。可以访问存储多个参考产物的数据库930。参考产物是如本文中所述的产物，并且可以对应于参考软件文件、参考设计模式或感兴趣的其他代码块。数据库可以被存储在很多位置，例如本地存储，或者存储在网络驱动器上，或者可以通过因特网或云访问，并且还可以跨多个存储设备分布。然后，可以通过将与程序片段对应的多个产物和与程序片段对应的多个参考产物相匹配来标识在一个或多个软件文件中的或与它们相关联的程序片段(例如接口故障)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。程序片段可以小至单个指令，或者与整个文件、程序、基本块、函数或接口一样大。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，该置信度对于某些实施例可以是可设置或可调整的，并且可以变化，例如上文关于标识文件所描述的。

[0117] 对于某些实施例，确定针对软件文件的产物包括将软件文件转换为中间表示，并且根据中间表示确定产物中的至少一个产物。对于某些实施例，软件文件和参考软件文件每个是源代码格式，或者每个是二进制代码格式。对于另外的实施例，程序片段对应于软件文件中的缺陷并且已经在数据库中被标识为对应于缺陷。另外的实施例可以自动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缺陷。某些实施例可以对修理选项排序，包括例如基于由用户选择的一个或多个先前修复选项，或者基于针对修复选项的成功的可能性。

[0118] 图10是示出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框图。示例系统包括可以与具有至少一个软件文件的源1010通信的接口1020。接口1020还通信地耦合到处理器1030。对于另外的实施例，接口1020还可以直接耦合到存储设备1040。该存储设备1040可以是各种各样公知的存储设备或系统，例如网络或本地存储设备，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可以存储参考产物，包括针对多个参考软件文件中的每个参考软件文件的参考产物，并且可以通信地耦合到处理器1030。处理器1030可以被配置为引起从源1010获取软件文件。该软件文件的身份以及该文件是否有较新版本可用、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还被配置为确定针对软件文件的多个产物，访问存储设备1040中的参考产物，将针对软件文件的产物与存储在存储设备1040中的参考产物进行比较，并且通过标识具有与比较的针对软件文件的产物对应的参考产物的参考软件文件来标识软件文件。

[0119] 在示例系统的另外的实施例中，处理器1030可以被配置为在存储设备1040中有可

用于文件的补丁时向软件文件自动应用补丁。在另外的实施例中，处理器还可以被配置为分析标识的补丁和软件文件以确定是否存在与软件文件中的缺陷的修复对应的补丁的修复部分，并且如果存在，则仅将补丁的修复部分自动应用于软件文件，或提示用户。

[0120] 图10的框图还可以示出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。该另一个示出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。对于另外的实施例，接口1020还可以直接耦合到存储设备1040。该存储设备1040可以是各种各样公知的存储设备或系统，例如网络或本地存储设备，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可以存储参考产物，并且可以通信地耦合到处理器1030。处理器1030可以被配置为引起一个或多个软件文件被获取，确定针对一个或多个软件文件的多个产物，访问存储多个参考产物的数据库，以及通过将与程序片段对应的多个产物和与程序片段对应的多个参考产物相匹配来标识针对一个或多个软件文件的程序片段。对于某些示例实施例，程序片段已经在数据库中被标识为对应于缺陷。这样的缺陷的示例包括故障、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相关。另外的实施例还可以使处理器被配置为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，程序片段在数据库中已经被标识为对应于特征，并且某些实施例还可以自动提供特征增强，包括以源代码或二进制文件的补丁的形式。

[0121] 修复

[0122] 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)与具有兼容接口(即，相同数目的参数、类型和输出)的元素可交换，并且可以通过用LLVM IR的修复块替换LLVM IR的缺陷块来转换LLVM IR。

[0123] 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代码。另外的实施例还可以当元素不存在时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。示例实施例可以利用所提取的图表示的层级，首先将层级升级到修复模式的合适表示，然后(经由编译)将层级降级到具体实现。产物的层级性质可以帮助形成修复代码。

[0124] 示例实施例可以使得用户能够提交目标程序(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)。用户可以选择针对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对未来的修复解决方案最佳地评级，并且也可以按评级顺序向用户呈现修复策略。某些实施例还可以自主运行，修复整个软件语料库上的缺陷或漏洞，包括连续地、周期性地和/或在设计环境中。

[0125] 除了上面讨论的实施例之外，本发明可以用于各种各样的用途。例如，可以在软件代码的编程期间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。另外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。又一其他示例实施例可以用于优化代码，包括标识未使用的代码、低效代码和建议代码以替换效率较低的代码。

[0126] 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。另外的实施例也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(例如故

障、安全漏洞和协议缺点)的认证。

[0127] 本发明的又一其他另外的示例实施例包括:代码重用发现器(寻找在代码库中执行相同事情的代码)、代码质量测量、对代码转变器的文本描述、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、故障/错误估计器、无用代码发现、代码特征映射、自动补丁审查器、代码改进决策工具(将特征列表映射到最小改变)、对现有设计工具(例如,企业架构)的扩展、替代实现建议器、代码探索和学习工具(例如,用于教学)、系统级代码许可证范围(footprint)和企业软件使用映射。

[0128] 应当理解,上述示例实施例可以用很多不同的方式来实现。在一些情况下,本文中描述的各种方法和机器可以都用具有中央处理器、存储器、盘或其他大容量存储、(多个)通信接口、(多个)输入/输出(I/O)设备和其他外围设备的物理、虚拟或混合通用计算机来实现。通用计算机被转换为执行上述方法的机器,例如通过将软件指令加载到数据处理器中,然后引起指令的执行以执行本文中描述的功能。软件指令还可以被模块化,例如具有用于摄取文件以形成语料库的摄取模块,用于确定针对语料库的文件和/或要被标识或分析用于设计模式的文件的产物的分析模块,用于执行机器学习的图分析模块和文本分析模块,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。对于某些示例实施例,这些模块可以组合或分离成另外的模块。

[0129] 如本领域中已知的,这种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部件之间进行数据传送的一组硬件线。总线是连接计算机系统的不同元件(例如,处理器、盘存储、存储器、输入/输出端口、网络端口等)的基本上共享的(多个)导管,其实现信息在元件之间传送。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。还连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)网络接口使得计算机能够连接到附接至网络的各种其他设备。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。盘或其他大容量存储为用于实现例如本文所述的各种过程的计算机软件指令和数据提供非易失性存储。

[0130] 因此,实施例通常可以用硬件、固件、软件或其任何组合来实现。此外,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他网络架构来访问。

[0131] 在某些实施例中,本文中描述的过程、设备和处理构成计算机程序产品,包括非暂态计算机可读介质,例如可移除存储介质,例如一个或多个DVD-ROM、CD-ROM、磁盘、磁带等,其提供用于系统的软件指令的至少部分。这样的计算机程序产品可以通过本领域公知的任何合适的软件安装过程来安装。在另一实施例中,软件指令的至少部分也可以通过线缆、通信和/或无线连接来下载。

[0132] 此外,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,本文中包含的这样的描述仅仅是为了方便,并且这样的动作实际上起因于计算设备、处理器、控制器或执行固件、软件、例程、指令等的其他设备。

[0133] 还应当理解,流程图、框图和网络图可以包括更多或更少的元件,这些元件不同地布置或者不同地表示。但是还应当理解,某些实现可以规定图示以特定方式实现的实施例的执行的块和网络图以及块和网络图的数目。

[0134] 因此，另外的实施例还可以用各种计算机体系结构、物理、虚拟、云计算机和/或其某种组合来实现，因此，本文中描述的数据处理器仅用于说明的目的，而非作为实施例的限制。

[0135] 虽然已经参考其示例实施例具体地示出和描述了本发明，但是本领域技术人员将理解，可以在不脱离由所附权利要求包括的本发明的范围的情况下在形式和细节方面做出各种变化。

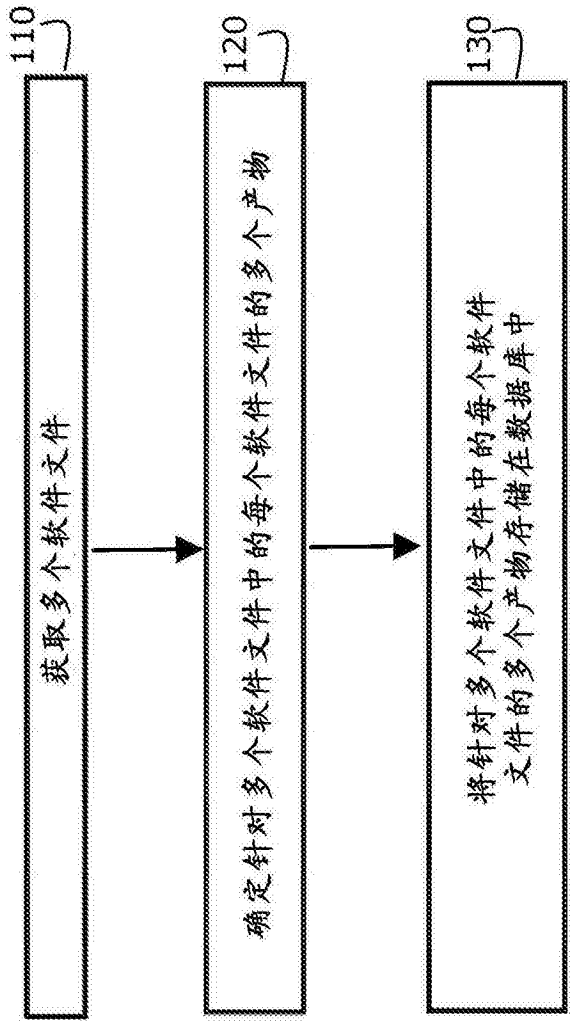

110
获取多个软件文件
120
确定针对多个软件文件中的每个软件文件的多个产物
130
将针对多个软件文件中的每个软件
文件的多个产物存储在数据库中

图1

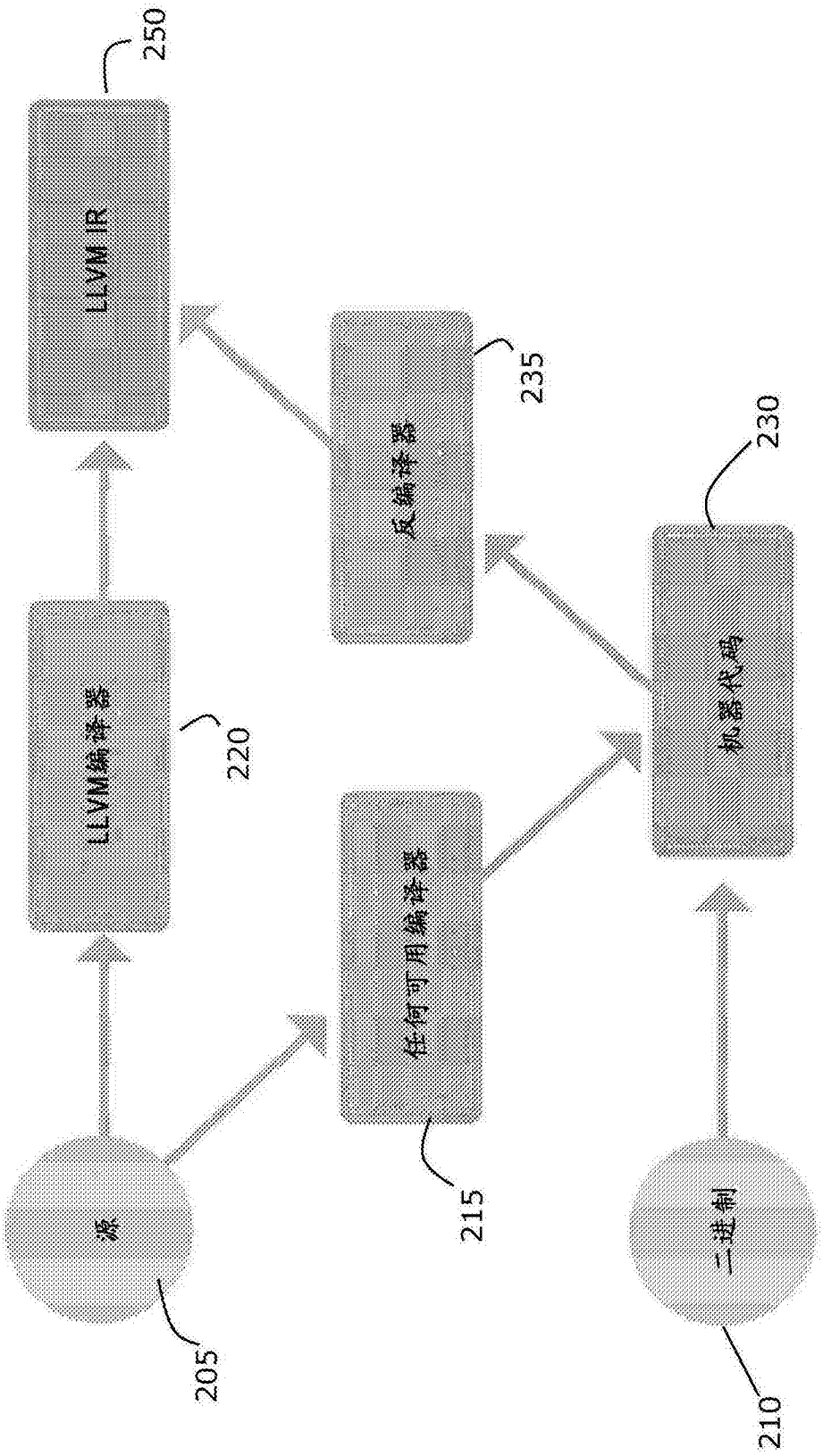

250
LLVM IR
235
反编译器
230
机器代码
LLVM编译器
220
任何可用编译器
源
205
215
二进制
210

图2

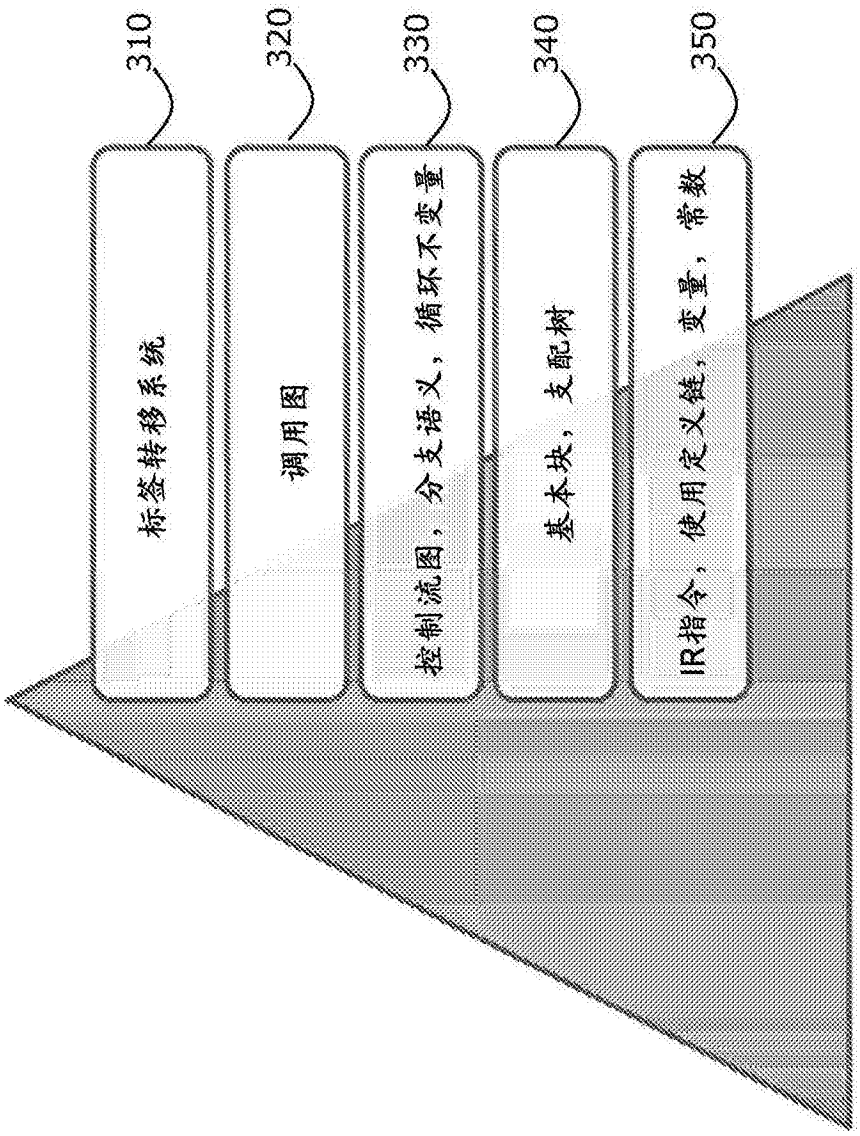

310
320
330
340
350
标签转移系统
调用图
控制流图，分支语义，循环不变量
基本块，支配树
IR指令，使用定义链，变量，常数

图3

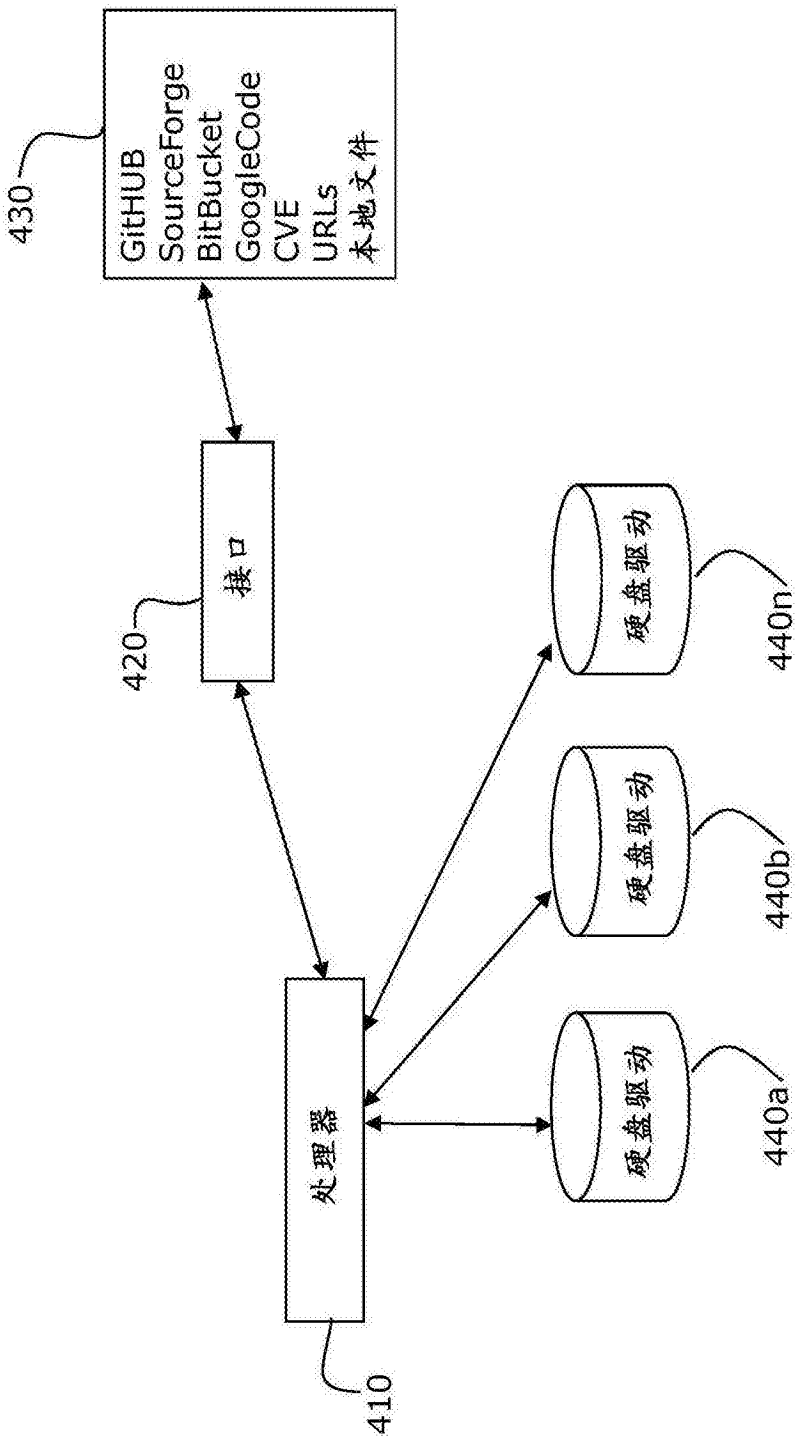

430
GitHUB
SourceForge
BitBucket
GoogleCode
CVE
URLs
本地文件
420
接口
处理器
410
硬盘驱动
440n
硬盘驱动
440b
硬盘驱动
440a

图4

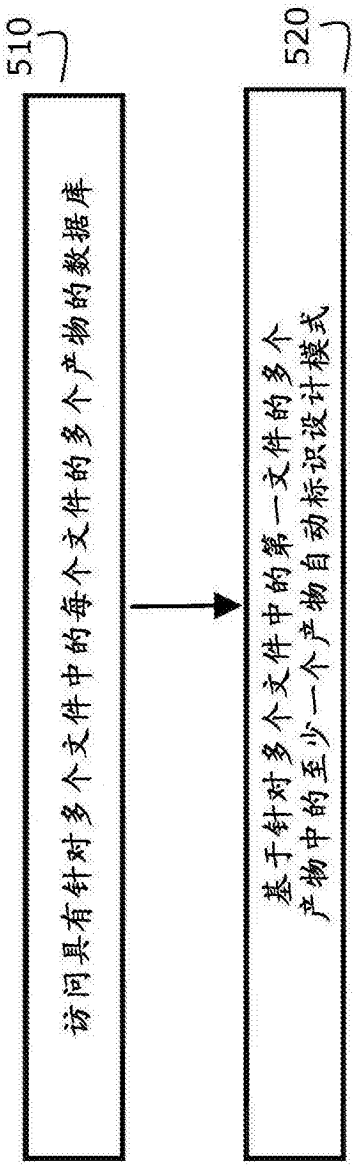

510
访问具有针对多个文件中的每个文件的多个产物的数据库
520
基于针对多个文件中的第一文件的多个产物中的至少一个产物自动标识设计模式

图5

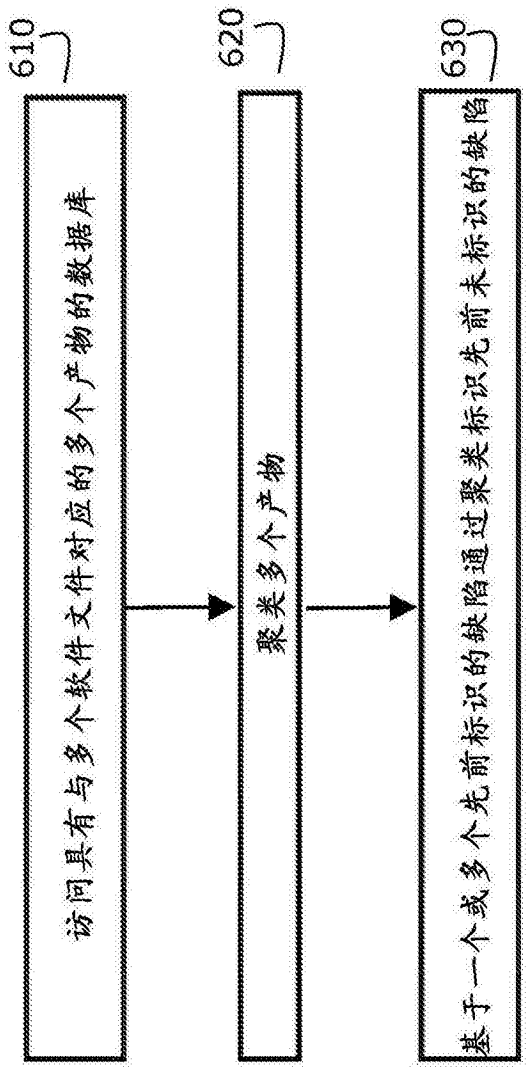

610
访问具有与多个软件文件对应的多个产物的数据库
620
聚类多个产物
630
基于一个或多个先前标识的缺陷通过聚类标识先前未标识的缺陷

图6

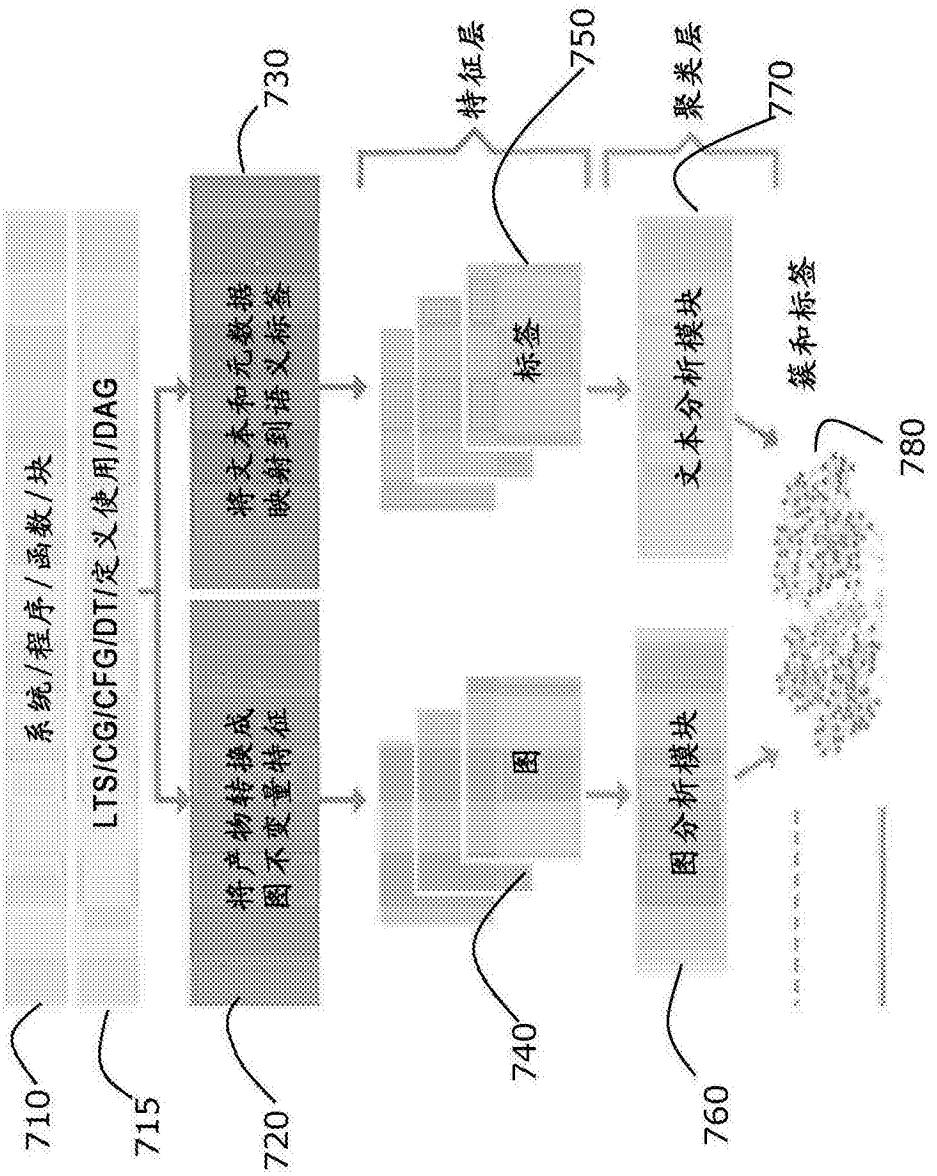

系统/程序/函数/块
LTS/CG/CFG/DT/定义使用/DAG
将产物转换成图不变量特征
将文本和元数据映射到语义标签
图
标签
特征层
聚类层
图分析模块
文本分析模块
簇和标签
710
715
720
730
740
750
760
770
780

图7

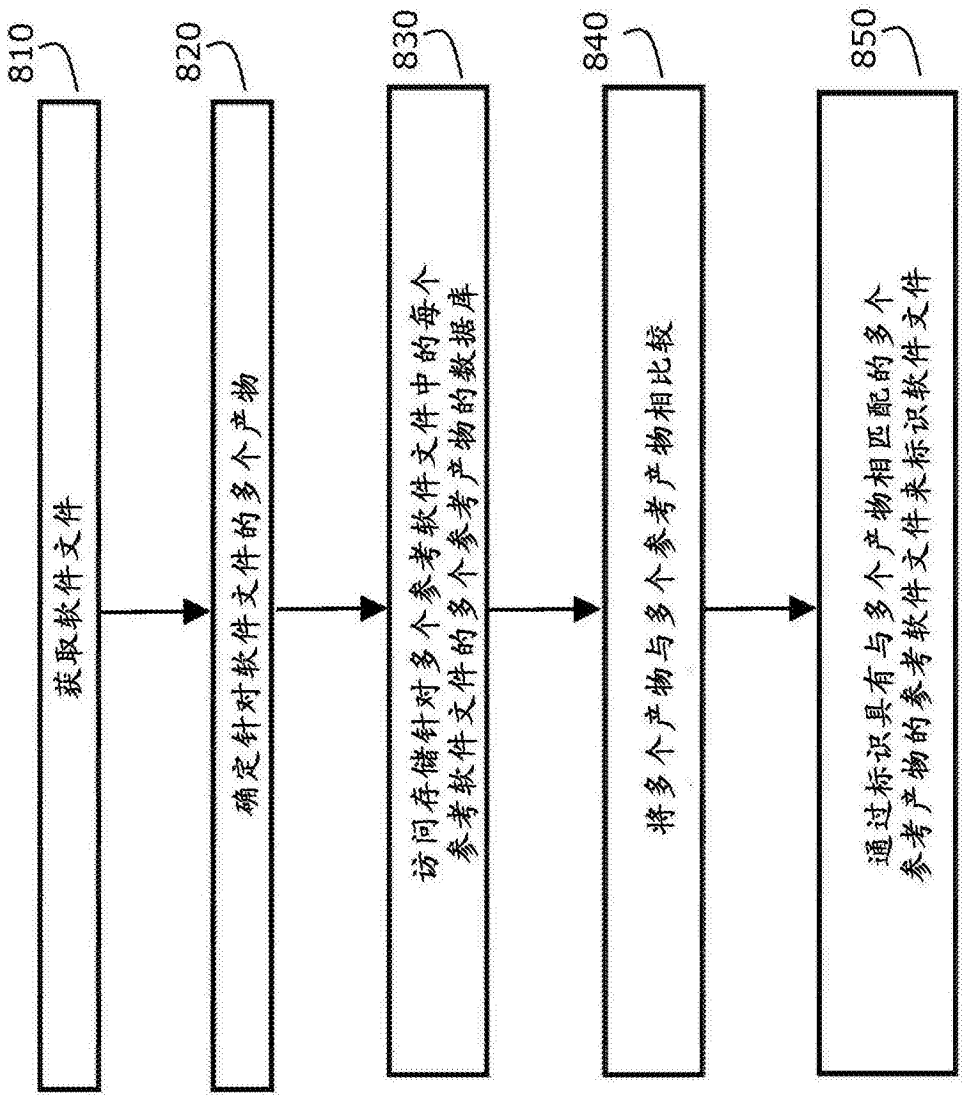

810
获取软件文件
820
确定针对软件文件的多个产物
830
访问存储针对多个参考软件文件中的每个参考软件文件的多个参考产物的数据库
840
将多个产物与多个参考产物相比较
850
通过标识具有与多个产物相匹配的多个参考产物的参考软件文件来标识软件文件

图8

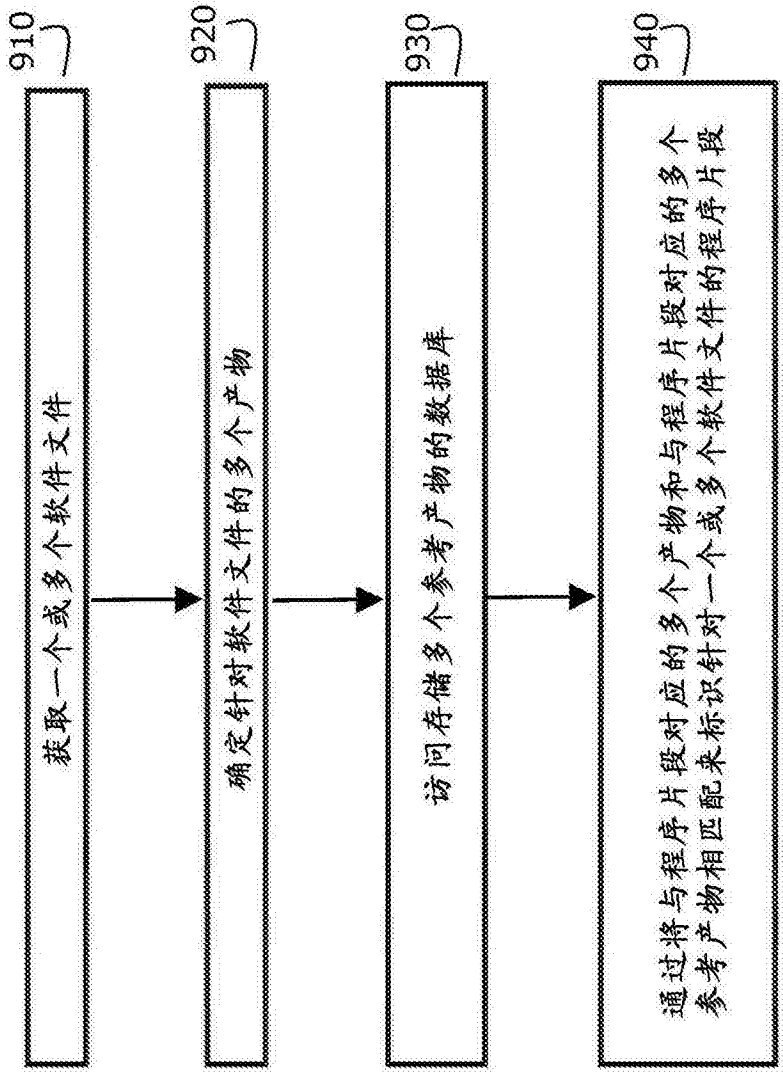

910
获取一个或多个软件文件
920
确定针对软件文件的多个产物
930
访问存储多个参考产物的数据库
940
通过将与程序片段对应的多个产物和与程序片段对应的多个参考产物相匹配来标识针对一个或多个软件文件的程序片段

图9

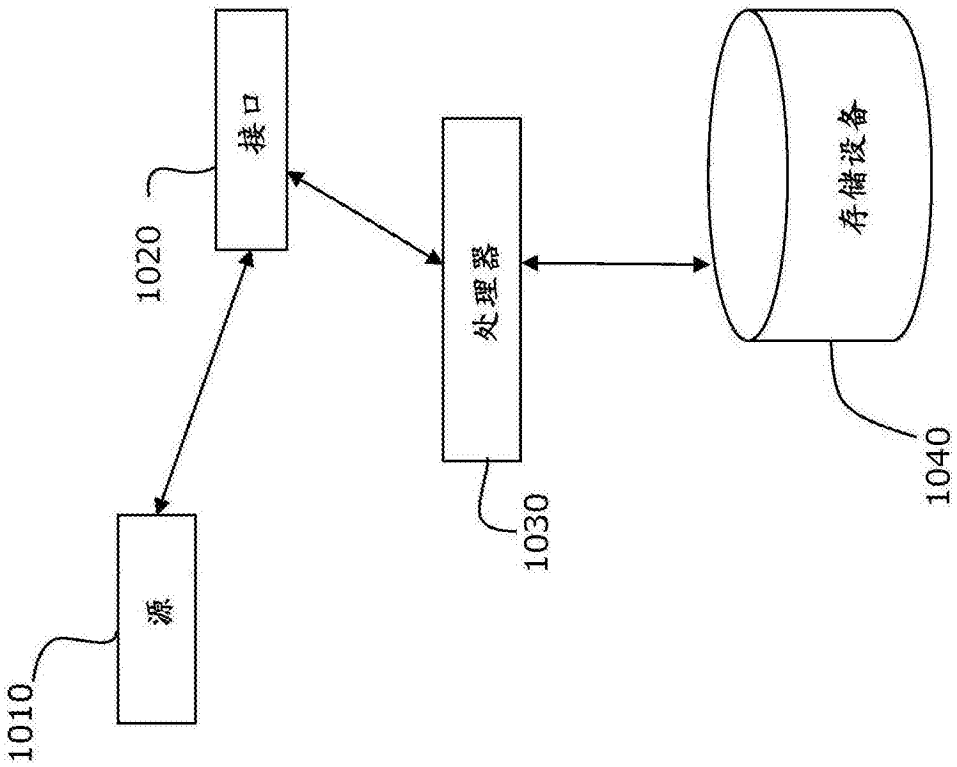


图10