(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 113688028 B
(45)授权公告日 2023.08.15

(21)申请号 202010425226.1

(22)申请日 2020.05.19

(65)同一申请的已公布的文献号
申请公布号 CN 113688028 A

(43)申请公布日 2021.11.23

(73)专利权人 成都鼎桥通信技术有限公司
地址 610041 四川省成都市高新区天华二路219号天府软件园C区3栋3-5层

(72)发明人 陈嘉

(74)专利代理机构 北京同立钧成知识产权代理有限公司 11205
专利代理师 张娜 臧建明

(51)Int.Cl.
G06F 11/36(2006.01)

(56)对比文件
CN 109683912 A,2019.04.26
CN 109240734 A,2019.01.18
WO 2017036335 A1,2017.03.09
US 2018121293 A1,2018.05.03
CN 109684215 A,2019.04.26
CN 107678773 A,2018.02.09
CN 105302716 A,2016.02.03
CN 109960643 A,2019.07.02
CN 105468507 A,2016.04.06
CN 111078274 A,2020.04.28
CN 107450933 A,2017.12.08
田江涛. 基于git工具的多分支并行开发上线流程.《电子技术与软件工程》.2019,第33页.

审查员 李宁

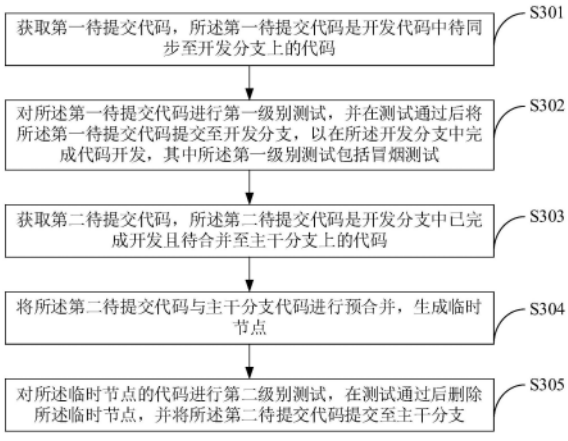
权利要求书2页 说明书11页 附图5页

(54)发明名称

代码提交方法和装置

(57)摘要

本发明实施例提供一种代码提交方法和装置,该方法包括:获取第一待提交代码,第一待提交代码是开发代码中待同步至开发分支上的代码;对第一待提交代码进行第一级别测试,并在测试通过后将第一待提交代码提交至开发分支,以在开发分支中完成代码开发,其中第一级别测试包括冒烟测试;获取第二待提交代码,第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码;将第二待提交代码与主干分支代码进行预合并,生成临时节点;对临时节点的代码进行第二级别测试,在测试通过后删除临时节点,并将所述第二待提交代码提交至主干分支。本发明实施例减少了因代码合入引起的持续构建中断次数和开发延迟时间,提高了代码提交质量和效率。



CN 113688028 B

1.一种代码提交方法,其特征在于,包括:

获取第一待提交代码,所述第一待提交代码是开发代码中待同步至开发分支上的代码;

对所述第一待提交代码进行第一级别测试,并在测试通过后将所述第一待提交代码提交至开发分支,以在所述开发分支中完成代码开发,其中所述第一级别测试包括冒烟测试,以检测开发分支上开发的代码对应编译生成的二进制组件的正确性;

获取第二待提交代码,所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码;

将所述第二待提交代码与主干分支代码进行预合并,生成临时节点;

对所述临时节点的代码进行第二级别测试,以检测主干分支整体代码的准确性;在测试通过后删除所述临时节点,并将所述第二待提交代码提交至主干分支;

所述对所述第一待提交代码进行第一级别测试,包括:

对所述第一待提交代码进行静态检查和编译,静态检查和编译并行开展;

对所述通过静态检查和编译的代码进行单元测试;

若所述第一待提交代码通过所述静态检查、编译和单元测试,则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中,将所述冒烟测试环境中对应的组件替换成所述二进制组件;

执行预先部署在所述冒烟测试环境中的冒烟测试用例,得到所述冒烟测试用例的执行数据和结果数据;

将所述执行数据和结果数据发送至终端,以指示目标人员根据所述执行数据和结果数据得到第一测试结果。

2.根据权利要求1所述的方法,其特征在于,所述获取第一待提交代码之前,所述方法还包括:

获取主干分支上的更新节点代码;

将所述更新节点代码同步至开发分支中。

3.根据权利要求1所述的方法,其特征在于,所述对所述临时节点的代码进行第二级别测试,包括:

对所述临时节点的代码进行静态检查和编译;

若所述临时节点的代码通过静态检查和编译,则对所述临时节点的代码进行开发者测试,得到开发者测试结果。

4.根据权利要求3所述的方法,其特征在于,所述方法还包括:

将所述第二待提交代码发送至终端,以指示目标人员对所述第二待提交代码进行人工审查;

获取人工审查结果;

根据所述人工审查结果和所述开发者测试结果,得到第二测试结果。

5.根据权利要求3或4所述的方法,其特征在于,所述对所述临时节点的代码进行开发者测试,包括:

对所述临时节点的代码进行单元测试和story流程测试。

6.一种代码提交装置,其特征在于,包括:

代码获取模块，用于获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码；

测试提交模块，用于对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试，以检测开发分支上开发的代码对应编译生成的二进制组件的正确性；

所述代码获取模块还用于：获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码；

预合并模块，用于将所述第二待提交代码与主干分支代码进行预合并，生成临时节点；

所述测试提交模块还用于：对所述临时节点的代码进行第二级别测试，以检测主干分支整体代码的准确性；在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支；

所述测试提交模块，具体用于：

对所述第一待提交代码进行静态检查和编译，静态检查和编译并行开展；

对所述通过静态检查和编译的代码进行单元测试；

若所述第一待提交代码通过所述静态检查、编译和单元测试，则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中，将所述冒烟测试环境中对应的组件替换成所述二进制组件；

执行预先部署在所述冒烟测试环境中的冒烟测试用例，得到所述冒烟测试用例的执行数据和结果数据；

将所述执行数据和结果数据发送至终端，以指示目标人员根据所述执行数据和结果数据得到第一测试结果。

7.一种电子设备，其特征在于，包括：至少一个处理器和存储器；

所述存储器存储计算机执行指令；

所述至少一个处理器执行所述存储器存储的计算机执行指令，使得所述至少一个处理器执行如权利要求1-5任一项所述的代码提交方法。

8.一种计算机可读存储介质，其特征在于，所述计算机可读存储介质中存储有计算机执行指令，当处理器执行所述计算机执行指令时，实现如权利要求1-5任一项所述的代码提交方法。

代码提交方法和装置

技术领域

[0001] 本发明实施例涉及计算机技术领域，尤其涉及一种代码提交方法和装置。

背景技术

[0002] 随着软件产品的开发复杂度不断增加，开发人员在开发过程中引入了持续构建和自动化测试来进行提交代码的质量防护。软件工程持续交付的原则之一便是要在构建流水线系统中快速提交代码、频繁构建以及建立尽量短的反馈路径。其中，快速提交有助于减少开发人员的等待时间，从而提升开发效率；频繁构建有助于缩小前后两个版本之间的差异，以便于查找问题并进行相应改进；尽量短的反馈路径有助于今早将问题暴露给代码提交人员，以尽快修复代码问题。

[0003] 相关技术中，提供了两种代码提交方法，一种是提供了一种流水线系统，通过该系统完整的实现了提交代码、频繁构建以及建立反馈路径这一开发过程，即把源代码配置、静态检查、软件部署和一系列自动化测试结合在一起，但是这种方法测试反馈时间长，仅靠基本功能测试就可以发现的问题无法提前暴露给提交人员，如果在系统集成测试时才发现代码问题，那么问题代码已经合入并污染了主干代码，为了解决该问题，开发人员需要在代码提交之前设计好手工验证用例，进行手工验证之后在提交代码。另一种是对待提交代码进行提前测试，即代码开发过程中，在代码提交前，对待提交分支代码进行测试和审核，达到预设测试及审核标准后，才允许分支代码合入主干分支。

[0004] 但是，第一种手工验证代码再提交的方式大大降低代码的提交效率，第二种方式中各开发分支上的待提交代码单独测试，即便测试通过，待提交代码提交后很容易与主干分支上最新节点的代码产生冲突影响，即代码合入主干分支后无法产生预期结果，出现持续构建中断的现象，降低了代码的提交质量和效率。

发明内容

[0005] 本发明实施例提供一种代码提交方法和装置，以解决现有技术中手工验证提交降低代码提交效率，以及只开发分支上的待提交代码单独测试，将测试后的开发分支的代码合入主干分支时无法产生预期效果降低代码提交质量效率的问题。

[0006] 本发明实施例的第一方面提供一种代码提交方法，包括：

[0007] 获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码；

[0008] 对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试；

[0009] 获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码；

[0010] 将所述第二待提交代码与主干分支代码进行预合并，生成临时节点；

[0011] 对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支。

[0012] 可选的，所述对所述第一待提交代码进行第一级别测试，包括：

[0013] 对所述第一待提交代码进行静态检查和编译；

[0014] 对所述通过静态检查和编译的代码进行单元测试；

[0015] 若所述第一待提交代码通过所述静态检查、编译和单元测试，则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中，对所述代码进行冒烟测试。

[0016] 可选的，所述获取第一待提交代码之前，所述方法还包括：

[0017] 获取主干分支上的更新节点代码；

[0018] 将所述更新节点代码同步至开发分支中。

[0019] 可选的，所述对所述代码进行冒烟测试，包括：

[0020] 将所述冒烟测试环境中对应的组件替换成所述二进制组件；

[0021] 执行预先部署在所述冒烟测试环境中的冒烟测试用例，得到所述冒烟测试用例的执行数据和结果数据；

[0022] 将所述执行数据和结果数据发送至终端，以指示目标人员根据所述执行数据和结果数据得到第一测试结果。

[0023] 可选的，所述对所述临时节点的代码进行第二级别测试，包括：

[0024] 对所述临时节点的代码进行静态检查和编译；

[0025] 若所述临时节点的代码通过静态检查和编译，则对所述临时节点的代码进行开发者测试，得到开发者测试结果。

[0026] 可选的，所述方法还包括：

[0027] 将所述第二待提交代码发送至终端，以指示目标人员对所述第二待提交代码进行人工审查；

[0028] 获取人工审查结果；

[0029] 根据所述人工审查结果和所述开发者测试结果，得到第二测试结果。

[0030] 可选的，所述对所述临时节点的代码进行开发者测试，包括：

[0031] 对所述临时节点的代码进行单元测试和story流程测试。

[0032] 本发明实施例的第二方面提供一种代码提交装置，包括：

[0033] 代码获取模块，用于获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码；

[0034] 测试提交模块，用于对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试；

[0035] 所述代码获取模块还用于：获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码；

[0036] 预合并模块，用于将所述第二待提交代码与主干分支代码进行预合并，生成临时节点；

[0037] 所述测试提交模块还用于：对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支。

[0038] 本发明实施例的第三方面提供一种电子设备，包括：至少一个处理器和存储器；

[0039] 所述存储器存储计算机执行指令；

[0040] 所述至少一个处理器执行所述存储器存储的计算机执行指令，使得所述至少一个处理器执行本发明实施例第一方面所述的代码提交方法。

[0041] 本发明实施例的第四方面提供一种计算机可读存储介质，所述计算机可读存储介质中存储有计算机执行指令，当处理器执行所述计算机执行指令时，实现本发明实施例第一方面所述的代码提交方法。

[0042] 本发明实施例提供一种代码提交方法和装置，通过获取第一待提交代码，其中第一待提交代码是开发代码中待同步至开发分支上的代码；然后对第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试；获取第二待提交代码，第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码；将所述第二待提交代码与主干分支代码进行预合并，生成临时节点；对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支。由于将整个代码测试流程分级进行代码测试，针对提交至开发分支的代码采用第一级别测试，且第一级别测试中包括冒烟测试，针对提交至主干分支的代码采用第二级别测试，避免了用户人工重复验证提交代码，提高了代码提交效率；同时，在将开发分支上开发完成的代码提交至主干分支之前先将其与主干代码进行预合并，生成临时节点，并对临时节点进行测试验证，可以及时发现开发分支代码合入主干分支后可能出现的冲突影响，在临时节点测试验证通过后删除临时节点，再将开发分支代码正式提交至主干分支，减少了因开发分支代码合入主干分支引起的代码冲突影响造成持续构建中断的次数，从而大大提高了代码提交的质量效率，以及提高了软件开发过程中持续构建的可靠性。

附图说明

[0043] 为了更清楚地说明本发明实施例或现有技术中的技术方案，下面将对实施例或现有技术描述中所需要使用的附图作简单地介绍，显而易见地，下面描述中的附图仅仅是本发明的一些实施例，对于本领域普通技术人员来讲，在不付出创造性劳动性的前提下，还可以根据这些附图获得其他的附图。

[0044] 图1是本发明一示例性实施例示出的代码分支开发流程示意图；

[0045] 图2是本发明一示例性实施例示出的代码分支开发节点示意图；

[0046] 图3是本发明一示例性实施例示出的代码提交方法的流程示意图；

[0047] 图4是本发明另一示例性实施例示出的提交代码测试方法的流程示意图；

[0048] 图5是本发明另一示例性实施例示出的提交代码测试方法的流程示意图；

[0049] 图6是本发明另一示例性实施例示出的提交代码测试方法的流程示意图；

[0050] 图7是本发明一示例性实施例示出的代码提交方法的应用场景图；

[0051] 图8是本发明一示例性实施例示出的代码提交装置的结构示意图；

[0052] 图9是本发明一示例性实施例示出的电子设备的结构示意图。

具体实施方式

[0053] 为使本发明实施例的目的、技术方案和优点更加清楚，下面将结合本发明实施例中的附图，对本发明实施例中的技术方案进行清楚、完整地描述，显然，所描述的实施例仅仅是本发明一部分实施例，而不是全部的实施例。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0054] 本发明的说明书和权利要求书及上述附图中的术语“第一”、“第二”、“第三”“第四”等(如果存在)是用于区别类似的对象，而不必用于描述特定的顺序或先后次序。应该理解这样使用的数据在适当情况下可以互换，以便这里描述的本发明的实施例例如能够以除了在这里图示或描述的那些以外的顺序实施。此外，术语“包括”和“具有”以及他们的任何变形，意图在于覆盖不排他的包含，例如，包含了一系列步骤或单元的过程、方法、系统、产品或设备不必限于清楚地列出的那些步骤或单元，而是可包括没有清楚地列出的或对于这些过程、方法、产品或设备固有的其它步骤或单元。

[0055] 随着软件产品的开发复杂度不断增加，开发人员在开发过程中引入了持续构建和自动化测试来进行提交代码的质量防护。软件工程持续交付的原则之一便是要在构建流水线系统中快速提交代码、频繁构建以及建立尽量短的反馈路径。其中，快速提交有助于减少开发人员的等待时间，从而提升开发效率；频繁构建有助于缩小前后两个版本之间的差异，以便于查找问题并进行相应改进；尽量短的反馈路径有助于今早将问题暴露给代码提交人员，以尽快修复代码问题。

[0056] 相关技术中，提供了两种代码提交方法，一种是提供了一种流水线系统，通过该系统完整的实现了提交代码、频繁构建以及建立反馈路径这一开发过程，即把源代码配置、静态检查、软件部署和一系列自动化测试结合在一起，但是这种方法测试反馈时间长，仅靠基本功能测试就可以发现的问题无法提前暴露给提交人员，如果在系统集成测试时才发现代码问题，那么问题代码已经合入并污染了主干代码。比如，假设提交一次代码在最后一步才测试失败，那么开发人员等待反馈时长则接近于整个代码提交流水线的时长，如果针对轻量级产品(比如独立网页应用程序Web APP)，使用该流水线系统来进行提交代码的适量防护测试不会出现大问题。然而，很多大型系统产品，比如嵌入式电信设备领域，本身在开发软件产品时就有比较复杂的版本构建方式，开发人员在开发、更新软件代码时，有可能仅仅生成一个新的二进制组件，但最终面向用户的是一个完成的系统安装包，如果二进制组件生成时间是10分钟，那么系统软件从打包到部署可能需要数小时，如果依然使用上述流水线系统对该系统软件代码进行防护测试，则会消耗很长时间。为了解决该问题，开发人员需要在代码提交之前设计好手工验证用例，执行验证用例通过后在提交增加或修改的代码。

[0057] 另一种代码提交方式是对待提交代码进行提前测试，即在分支代码开发过程中，在开发出来的代码提交前，对待提交分支代码进行测试和审核，达到预设测试及审核标准后，才允许分支代码合入主干分支。如图1所示的分支开发流程示意图，以基于GitLab代码管理系统工作为例，一个软件产品(软件版本)对应存在一个主干分支，产品的持续构建基于主干分支代码进行，当迭代过程中有多个特性要并行开发时，由开发各自拉出自己的开发分支(比如图1中的第一开发分支和第二开发分支)进行独立开发，在特性开发完毕后，再向主干分支发起合入请求(记为Merge，简记为MR)，在每个开发分支上的代码合入主干分支前，对每个开发分支上开发完的待合入代码(或称为待提交代码)进行测试和审核，测试审

核通过后再提交至主干分支，最终正式的产品版本会通过主干分支上的代码进行构建发布。

[0058] 但是，第一种手工验证代码再提交的方式大大降低代码的提交效率，第二种方式中各开发分支上的待提交代码单独测试，即便测试通过，待提交代码提交后很容易与主干分支上最新节点的代码产生冲突影响，即代码合入主干分支后无法产生预期结果，出现持续构建中断的现象，降低了代码的提交质量和效率。比如，参见图2，主干分支上有四个节点，分别为第一节点1、第二节点2、第三节点3和第四节点4，主干分支在第一节点1拉出第一开发分支，开发人员在第一开发分支的代码演进过程中定义了一个新的全局变量，而主干分支代码在向第二节点2演进过程中也增加了一个同名的全局变量，当第一开发分支代码向主干分支发起MR时，即便开发人员在代码合入前对第一开发分支代码进行了测试和审核，且测试和审核均通过，一旦第一开发分支代码合入主干分支生成第三节点3，基于主干分支的构建立即会产生因变量重复定义导致的编译问题，即出现构建中断。再比如，主干分支在第二节点2拉出第二开发分支，第二开发分支开发完毕后向主干分支发起MR生成第四节点4，也有可能造成第二开发分支代码合入主干分支后，对后续主干分支代码的运行逻辑造成影响，也会导致构建中断。

[0059] 针对此缺陷，本发明的技术方案主要在于：将整个代码测试流程分级进行代码测试，针对提交至开发分支的代码采用第一级别测试，且第一级别测试中包括冒烟测试，主要检测开发分支上开发的代码对应编译生成的二进制组件的正确性，针对提交至主干分支的代码采用第二级别测试，主要检测主干分支整体代码的准确性，避免了用户人工重复验证提交代码，提高了代码提交效率；同时，在将开发分支上开发完成的代码提交至主干分支之前先将其与主干代码进行预合并，生成临时节点，并对临时节点进行测试验证，可以及时发现开发分支代码合入主干分支后可能出现的冲突影响，在临时节点测试验证通过后删除临时节点，再将开发分支代码正式提交至主干分支，减少了因开发分支代码合入主干分支引起的代码冲突影响造成持续构建中断的次数，从而大大提高了代码提交的质量效率，以及提高了软件开发过程中持续构建的可靠性。

[0060] 图3是本发明一示例性实施例示出的代码提交方法的流程示意图。

[0061] 如图3所示，本实施例提供的方法主要包括以下步骤。

[0062] S301，获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码。

[0063] 具体的，在基于主干分支开发过程中，拉出开发分支后，开发人员根据需求在本地进行开发，生成一系列开发代码，这些开发代码在开发分支上进行开发演进，开发分支上的开发完成的需要提交至主干分支，与主干分支代码合并，基于最终的主干分支代码进行持续构建，最终生成正式的软件产品或软件版本。

[0064] S302，对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试。

[0065] 具体的，在开发分支上开发完成的代码提交至主干分支前，需要对待提交代码进行质量防护测试，以保证开发分支代码的正确性。

[0066] 本步骤中，第一级别测试可以时个人级自动防护测试，并且，将冒烟测试纳入个人

级防护测试，能够自动检测开发人员在开发分支上开发的代码的正确性，避免了开发人员反复以手工验证用例的方式对开发分支上开发的代码进行测试。

[0067] S303，获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码。

[0068] 本步骤中，在开发分支中将待提交至主干分支的代码签出，得到第二待提交代码。

[0069] S304，将所述第二待提交代码与主干分支代码进行预合并，生成临时节点。

[0070] 具体的，在第二待提交代码正式提交至主干分支与主干分支代码合并前，先将第二待提交代码与主干分支代码进行预合并，并在主干分支上生成临时节点。

[0071] S305，对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支。

[0072] 具体的，对临时节点的代码进行系统级防护测试，以检测主干分支代码的健康性，临时节点的测试效果等同于正式合并后的测试效果，因此，若临时节点的代码正确，则说明开发分支上的代码正式提交至主干分支与主干分支代码合并后不会造成主干分支代码运行逻辑错误，也不会造成因同名变量引起的同名变量编译错误，从而也不会造成基于主干分支代码持续构建中断。

[0073] 本实施例中，通过将整个代码测试流程分级进行代码测试，针对提交至开发分支的代码采用第一级别测试，且第一级别测试中包括冒烟测试，主要检测开发分支上开发的代码对应编译生成的二进制组件的正确性，针对提交至主干分支的代码采用第二级别测试，主要检测主干分支整体代码的准确性，避免了用户人工重复验证提交代码，提高了代码提交效率；同时，在将开发分支上开发完成的代码提交至主干分支之前先将其与主干代码进行预合并，生成临时节点，并对临时节点进行测试验证，可以及时发现开发分支代码合入主干分支后可能出现的冲突影响，在临时节点测试验证通过后删除临时节点，再将开发分支代码正式提交至主干分支，减少了因开发分支代码合入主干分支引起的代码冲突影响造成持续构建中断的次数，从而大大提高了代码提交的质量效率，以及提高了软件开发过程中持续构建的可靠性。

[0074] 图4是本发明另一示例性实施例示出的提交代码测试方法的流程示意图，本实施例主要对提交至开发分支的代码进行第一级别测试的过程进行详细描述。

[0075] 需要说明的是，针对待提交至开发分支的代码，本实施例采用的是个人级防护测试。

[0076] 如图4所示，本实施例提供的方法主要包括以下步骤。

[0077] S401，对所述第一待提交代码进行静态检查和编译。

[0078] 本步骤中，静态检查可以但不限于是PC-Lint、Fortify、Coverity等，静态检查和编译可并行开展，以缩短测试时间；其中对第一待提交代码进行编译生成的是二进制组件。

[0079] 在一个实施例中，在对第一待提交代码进行静态检查之前和编译之前，还包括：将分支更新节点代码更新到本地。

[0080] 具体的，一个主干分支可能拉出多个开发节点，需要多个开发人员协同开发，在多人协同开发的场景中，通过将每个开发分支上开发的代码均同步至本地，可以避免多个开发分支向主干分支提交代码时发生代码提交冲突，如果在更新合并过程中发生冲突，则测试流程自动停止，待开发人员解决代码冲突后重新发起测试流程。

[0081] S402,对所述第一待提交代码进行单元测试。

[0082] 具体的,对第一待提交代码进行自动化单元测试,该测试不依赖于真实用户环境,可以在类开发环境中执行,主要关注函数级以及部分模块的执行逻辑正确性,一些对真实环境或者其他组件对象的依赖可以通过Mock的方式模拟。

[0083] S403,若所述第一待提交代码通过所述静态检查、编译和单元测试,则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中,对所述代码进行冒烟测试。

[0084] 本步骤中,如果第一待提交代码编译正常,且通过了静态检查和单元测试,则将步骤S401中编译生成的二进制组件加载至冒烟测试环境中。

[0085] 具体的,将所述冒烟测试环境中对应的组件替换成所述二进制组件;执行预先部署在所述冒烟测试环境中的冒烟测试用例,得到所述冒烟测试用例的执行数据和结果数据;将所述执行数据和结果数据发送至终端,以指示目标人员根据所述执行数据和结果数据得到第一测试结果。其中,冒烟测试用例是预先确认有效的用例。

[0086] 本实施例中,冒烟测试用例相对稳定,且过程短小,测试过程中可减少测试时间。同时,通过将编译生成的二进制组件下载到冒烟测试环境并完成组件替换激活,而不是对完整的软件包进行测试,减少了软件打包和软件包下载的时间消耗,尽量通过最小可替换单元(二进制组件)完成冒烟测试,大大节省了个人防护测试阶段的时间,提高了测试质量和效率。

[0087] 图5是本发明另一示例性实施例示出的提交代码测试方法的流程示意图,本实施例主要对开发分支待提交至主干分支的代码进行第二级别测试的过程进行详细描述。

[0088] 需要说明的是,针对开发分支待提交至主干分支的代码,本实施例中采用的是系统级质量防护测试。

[0089] 如图5所示,本实施例提供的方法可以包括以下步骤。

[0090] S5010,接受开发人员发送的合并指令,所述合并指令用于出发第二待提交代码合并至主干分支代码。

[0091] 具体的,当一个开发分支开发完成,需要进入主干分支时,由开发人员发起合入请求(Merge Request,MR),通过MR触发后续一系列系统级防护测试流程。

[0092] S5021,将所述第二待提交代码与主干分支代码进行预合并,生成临时节点。

[0093] 具体的,构建服务器将开发分支代码和主干分支代码进行一次合并,生成临时节点,此合并过程中如发生合并冲突,流程自动停止,待开发人员在待提交开发分支上解决冲突之后,再重新发起MR。

[0094] S5022,对所述临时节点的代码进行静态检查和编译;若所述临时节点的代码通过静态检查和编译,则对所述临时节点的代码进行开发者测试,得到开发者测试结果。

[0095] 本步骤中,静态检查和编译与上述对第一代提交代码进行静态检查和编译的过程一致,此处不再重复说明。

[0096] 通过静态检查和编译后,则继续进行开发者测试,包括对所述临时节点的代码进行单元测试以及对story流程测试。测试过程尽量不依赖桩,以接近真实系统运行流程。

[0097] 进一步的,静态检查、开发者测试均通过后,则自动删除临时节点。

[0098] S5031,将所述第二待提交代码发送至终端,以指示目标人员对所述第二待提交代

码进行人工审查。

[0099] S5032,获取人工审查结果。

[0100] S5040,根据所述人工审查结果和所述开发者测试结果,得到第二测试结果,并在所述第二测试结果通过时将所述第二待提交代码提交至主干分支。

[0101] 具体的,在步骤S5021和S5022中完成测试且测试通过后,删除临时节点,待人工审查通过后,则自动将第二待提交代码提交至主干分支。

[0102] 本实施例中,在开发分支代码正式合入主干分支代码之前,先进行预合并,并生成临时节点,对临时节点的代码进行测试,由于临时节点的测试效果等同于正式合并后的测试效果,因此,若临时节点的代码正确,则说明开发分支上的代码正式提交至主干分支与主干分支代码合并后不会造成主干分支代码运行逻辑错误,也不会造成因同名变量引起的同名变量编译错误,从而也不会造成基于主干分支代码持续构建中断。因此,通过预合并可以及时发现开发分支代码合入主干分支后可能出现的冲突影响,在临时节点测试验证通过后删除临时节点,再将开发分支代码正式提交至主干分支,减少了因开发分支代码合入主干分支引起的代码冲突影响造成持续构建中断的次数,从而大大提高了代码提交的质量效率,以及提高了软件开发过程中持续构建的可靠性。

[0103] 图6是本发明另一示例性实施例示出的提交代码测试方法的流程示意图,本实施例对代码提交的整个质量防护测试流程进行描述。

[0104] 如图6所示,本实施例提供的方法可以包括以下步骤。

[0105] S601,将主干分支上更新节点代码同步至开发分支。

[0106] 具体的,在基于分支开发的过程中,开发分支拉出后,开发人员周期将主干分支上的最新节点代码同步到开发分支,如此可以尽量缩小开发分支与主干分支的差异,避免开发分支代码合入主干分支代码时产生大量冲突,影响代码提交质量和效率。

[0107] S602,获取第一待提交代码,所述第一待提交代码是开发代码中待同步至开发分支上的代码,对所述第一待提交代码进行测试,并在测试通过后将所述第一待提交代码提交至开发分支,以在所述开发分支上完成代码开发。

[0108] 本步骤的具体实现方法可参考图4所示实施例中的代码测试方法,此处不再重复说明。

[0109] S603,获取第二待提交代码,所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码,对所述第二待提交代码进行测试,并在测试通过后将所述第二待提交代码提交至主干分支。

[0110] 本步骤的具体实现方法可参考图5所示实施例中的代码测试方法,此处不再重复说明。

[0111] S604,基于主干分支上的代码进行持续构建,以生成软件产品。

[0112] 具体的,在所有开发人员在各自的开发分支上完成代码开发之后,将所有开发分支上开发的代码提交至主干分支,然后基于主干分支代码进行持续构建,最终生成正式的软件产品或软件版本。

[0113] 通过本实施例提供的方法,因开发人员代码修改导致的主干构建状态不健康或阻塞的情况大大降低,减少了因开发分支代码合入主干分支代码引起的持续构建中断次数,减少了测试时间,大大提升了代码提交的质量效率和持续构建的可靠性,从而提高了开发

的软件产品的稳定性。

[0114] 图7是本发明一示例性实施例示出的代码提交方法的应用场景图。

[0115] 如图7所示，本实施例提供的应用场景的基本架构主要包括：中心服务器(Master)701，构建执行机节点(Slave)702以及GitLab与Git代码仓库703。

[0116] 其中，构建执行机节点包括：静态检查Slave，编译Slave，单元测试Slave，代码工程源机，以及冒烟测试环境。

[0117] 其中，中心服务器(Master)，主要负责接收开发者或钩子行为的触发构建请求、管理构建资源、构建任务的调度和分发、构建结果的汇总上报。个人防护构建和系统防护构建的主要ant脚本都可以放在Master上；构建执行机节点(Slave)，用于具体的构建任务执行。不同的Slave可并行执行不同的任务以加快构建速度。在图7所示方案中，个人配置好的冒烟测试环境也算作执行机节点Slave，用于被Master指派执行开发者自定义任务。其中，项目代码工程源机，用于定期(如每日一次)从主干分支下载代码并向各个构建Slave同步这些公共代码(包括构建依赖的其他组件或者库)，这样开发者在构建时只需同步增量代码，缩短了拷贝时间；GitLab与Git代码仓库，用于源代码管理、提交审核以及注册合并请求的钩子行为。

[0118] 结合图7所示的应用场景图对整个代码提交方法进行详细描述如下：

[0119] 第一步，开发人员在桌面完成代码改动之后，手动(脚本或页面)触发远程Master开始一次个人级防护构建，Master根据当前Slave节点的可用状态，首先将开发者本地的增量修改代码用rsync工具同步到各节点上，然后将代码的静态检查和编译任务分发到对应节点，并触发任务的执行。

[0120] 第二步，编译和静态检查任务执行通过，Master顺序触发单元测试任务对代码进行单元测试。

[0121] 第三步，单元测试任务执行通过后，如果开发者配置了个人冒烟的Job，则触发个人冒烟测试，Master将编译Slave上编译出来的二进制组件同步到个人冒烟环境，并触发后继组件下载替换、用例执行等一系列步骤，已完成冒烟测试。

[0122] 第四步，个人冒烟用例执行通过，各Slave上的执行结果以网页(html)形式返回给Master，Master汇总后以邮件或页面形式发送终端以通知开发者，Master同时修改开发者门禁标志位，允许pre-commit钩子通过检查，即代码获得提交权限，将代码提交至开发分支。

[0123] 第五步，分支开发完成后，开发者在GitLab上发起合入请求MR，该MR事件会触发对应的webhook钩子行为，通知Jenkins Master开始进行系统级防护构建(已提前设定只有Jenkins流水线运行成功时才允许在GitLab界面上合并)。

[0124] 第六步，Jenkins Master被触发系统级防护构建后，首先将待提交分支代码签出，与目标分支(主干分支)进行一次预合并操作，预合并在Master工作路径下的本地主干分支上生成新的临时节点，然后再通过rsync工具向Slave进行增量代码同步，并触发后继测试任务。

[0125] 第七步，后继测试过程类似于个人级防护构建，不再赘述，系统级构建防护并不执行冒烟用例，因为主干上的每日定时滚动构建会去执行正式的冒烟测试。在系统级构建完成后，只需要代码审查通过，GitLab即可允许MR内容被自动合入主干分支。

[0126] 图8是本发明一示例性实施例示出的代码提交装置的结构示意图。

[0127] 如图8所示，本实施例提供的装置包括：代码获取模块801，测试提交模块802和预合并模块803；其中，代码获取模块，用于获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码；测试提交模块，用于对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试；所述代码获取模块还用于：获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码；预合并模块，用于将所述第二待提交代码与主干分支代码进行预合并，生成临时节点；所述测试提交模块还用于：对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支。

[0128] 进一步的，所述测试提交模块具体用于：对所述第一待提交代码进行静态检查和编译；对所述通过静态检查和编译的代码进行单元测试；若所述第一待提交代码通过所述静态检查、编译和单元测试，则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中，对所述代码进行冒烟测试。

[0129] 进一步的，所述代码获取模块还用于：获取主干分支上的更新节点代码；将所述更新节点代码同步至开发分支中。

[0130] 进一步的，所述测试提交模块具体用于：将所述冒烟测试环境中对应的组件替换成所述二进制组件；执行预先部署在所述冒烟测试环境中的冒烟测试用例，得到所述冒烟测试用例的执行数据和结果数据；将所述执行数据和结果数据发送至终端，以指示目标人员根据所述执行数据和结果数据得到第一测试结果。

[0131] 进一步的，所述测试提交模块具体用于：对所述临时节点的代码进行静态检查和编译；若所述临时节点的代码通过静态检查和编译，则对所述临时节点的代码进行开发者测试，得到开发者测试结果。

[0132] 进一步的，所述测试提交模块具体用于：将所述第二待提交代码发送至终端，以指示目标人员对所述第二待提交代码进行人工审查；获取人工审查结果；根据所述人工审查结果和所述开发者测试结果，得到第二测试结果。

[0133] 进一步的，所述测试提交模块具体用于：对所述临时节点的代码进行单元测试和story流程测试。

[0134] 本实施例中各个模块的具体功能实现可参考上述有关方法实施例中的描述，此处不再赘述。

[0135] 图9为本发明实施例提供的电子设备的硬件结构示意图。如图9所示，本实施例提供的电子设备90包括：至少一个处理器901和存储器902。其中，处理器901、存储器902通过总线903连接。

[0136] 在具体实现过程中，至少一个处理器901执行所述存储器902存储的计算机执行指令，使得至少一个处理器901执行上述方法实施例中的代码提交方法。

[0137] 处理器901的具体实现过程可参见上述方法实施例，其实现原理和技术效果类似，本实施例此处不再赘述。

[0138] 在上述的图9所示的实施例中，应理解，处理器可以是中央处理单元(英文：Central Processing Unit，简称：CPU)，还可以是其他通用处理器、数字信号处理器(英文：

Digital Signal Processor,简称:DSP)、专用集成电路(英文:Application Specific Integrated Circuit,简称:ASIC)等。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。结合发明所公开的方法的步骤可以直接体现为硬件处理器执行完成,或者用处理器中的硬件及软件模块组合执行完成。

[0139] 存储器可能包含高速RAM存储器,也可能还包括非易失性存储NVM,例如至少一个磁盘存储器。

[0140] 总线可以是工业标准体系结构(Industry Standard Architecture,ISA)总线、外部设备互连(Peripheral Component Interconnect,PCI)总线或扩展工业标准体系结构(Extended Industry Standard Architecture,EISA)总线等。总线可以分为地址总线、数据总线、控制总线等。为便于表示,本申请附图中的总线并不限定仅有一根总线或一种类型的总线。

[0141] 本申请的另一实施例提供一种计算机可读存储介质,所述计算机可读存储介质中存储有计算机执行指令,当处理器执行所述计算机执行指令时,实现上述方法实施例中的代码提交方法。

[0142] 上述的计算机可读存储介质,上述可读存储介质可以是由任何类型的易失性或非易失性存储设备或者它们的组合实现,如静态随机存取存储器(SRAM),电可擦除可编程只读存储器(EEPROM),可擦除可编程只读存储器(EPROM),可编程只读存储器(PROM),只读存储器(ROM),磁存储器,快闪存储器,磁盘或光盘。可读存储介质可以是通用或专用计算机能够存取的任何可用介质。

[0143] 一种示例性的可读存储介质耦合至处理器,从而使处理器能够从该可读存储介质读取信息,且可向该可读存储介质写入信息。当然,可读存储介质也可以是处理器的组成部分。处理器和可读存储介质可以位于专用集成电路(Application Specific Integrated Circuits,简称:ASIC)中。当然,处理器和可读存储介质也可以作为分立组件存在于设备中。

[0144] 本领域普通技术人员可以理解:实现上述各方法实施例的全部或部分步骤可以通过程序指令相关的硬件来完成。前述的程序可以存储于一计算机可读取存储介质中。该程序在执行时,执行包括上述各方法实施例的步骤;而前述的存储介质包括:ROM、RAM、磁碟或者光盘等各种可以存储程序代码的介质。

[0145] 最后应说明的是:以上各实施例仅用以说明本发明的技术方案,而非对其限制;尽管参照前述各实施例对本发明进行了详细的说明,本领域的普通技术人员应当理解:其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分或者全部技术特征进行等同替换;而这些修改或者替换,并不使相应技术方案的本质脱离本发明各实施例技术方案的范围。

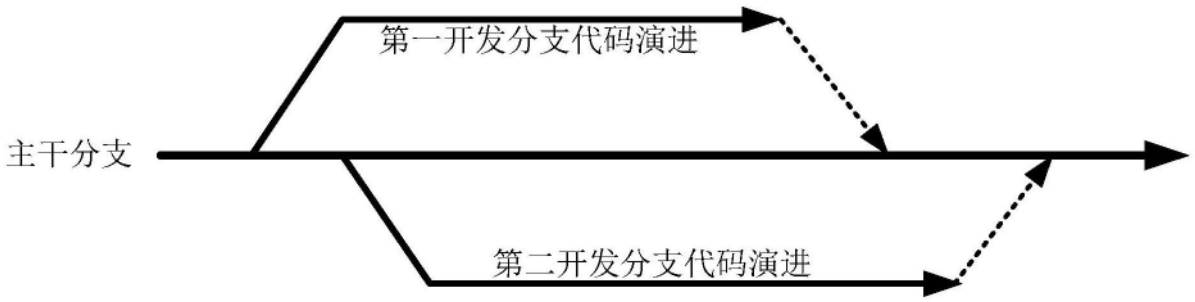

第一开发分支代码演进
主干分支
第二开发分支代码演进

图1

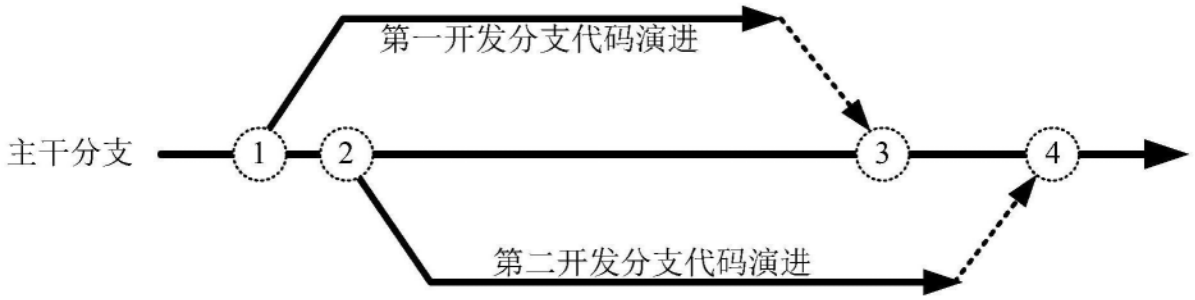

第一开发分支代码演进
主干分支
1
2
3
4
第二开发分支代码演进

图2

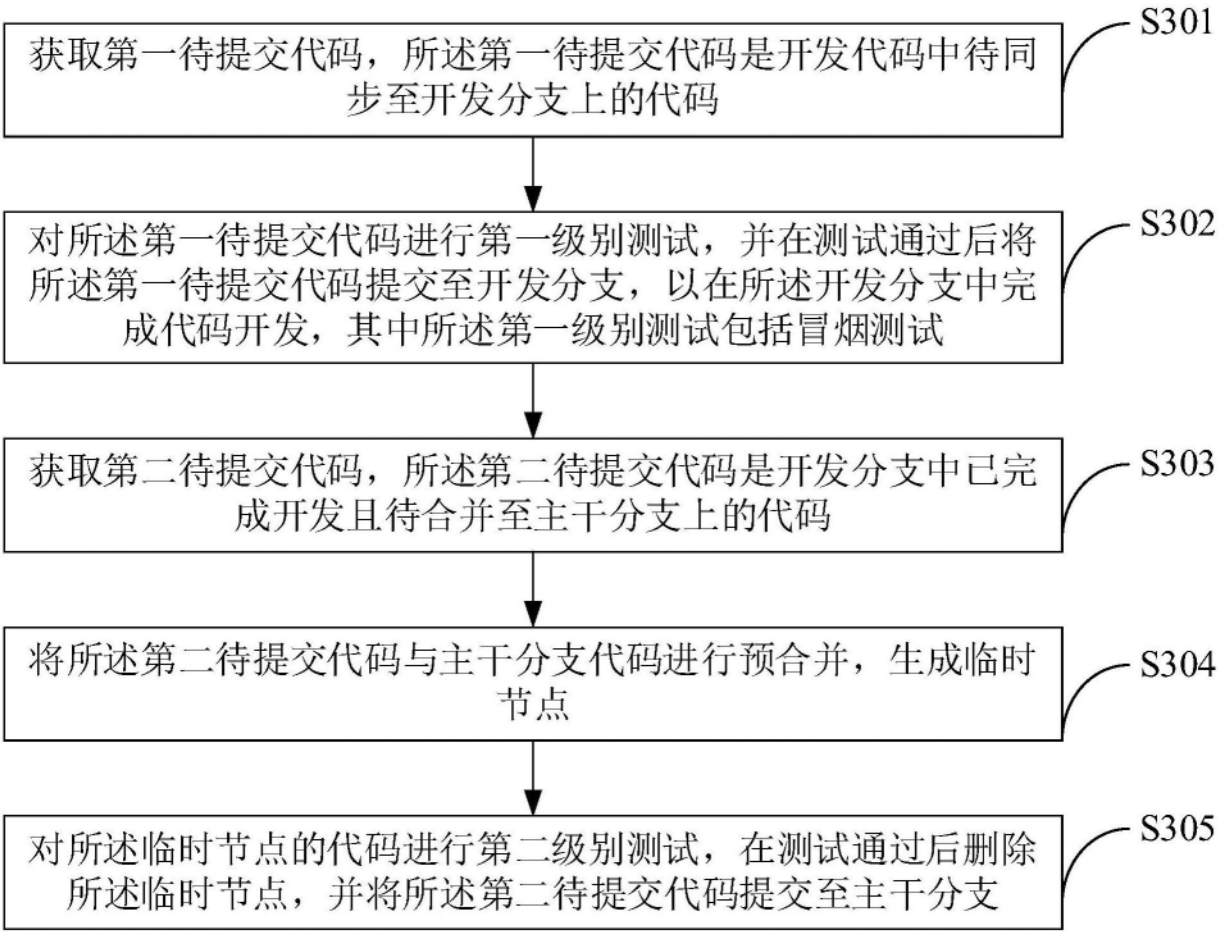

获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码
S301
对所述第一待提交代码进行第一级别测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支中完成代码开发，其中所述第一级别测试包括冒烟测试
S302
获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码
S303
将所述第二待提交代码与主干分支代码进行预合并，生成临时节点
S304
对所述临时节点的代码进行第二级别测试，在测试通过后删除所述临时节点，并将所述第二待提交代码提交至主干分支
S305

图3

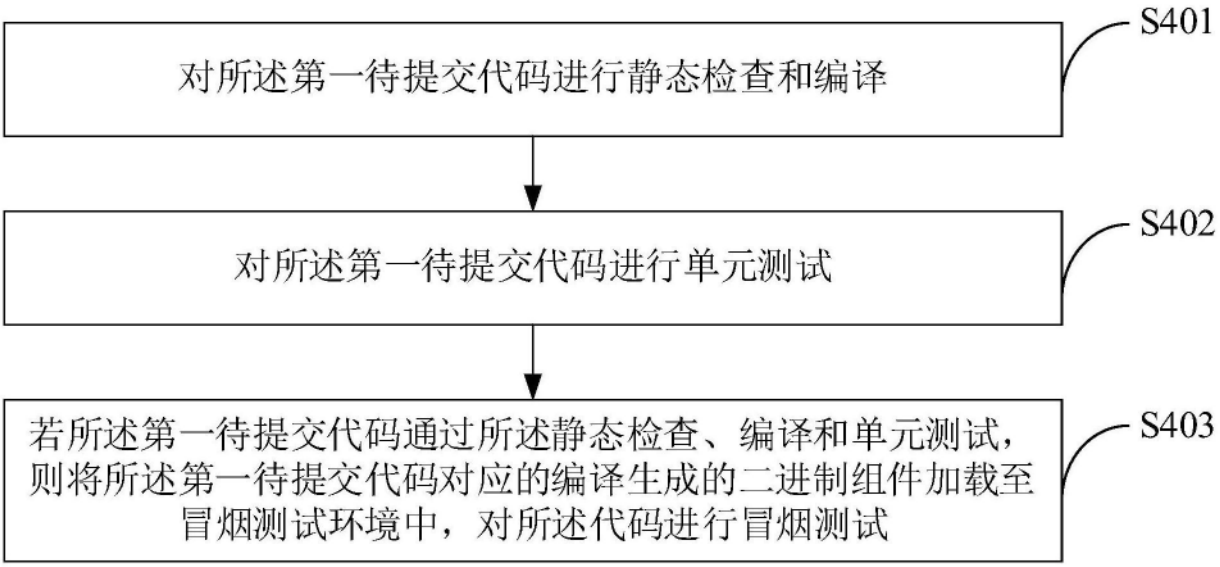

对所述第一待提交代码进行静态检查和编译
S401
对所述第一待提交代码进行单元测试
S402
若所述第一待提交代码通过所述静态检查、编译和单元测试，则将所述第一待提交代码对应的编译生成的二进制组件加载至冒烟测试环境中，对所述代码进行冒烟测试
S403

图4

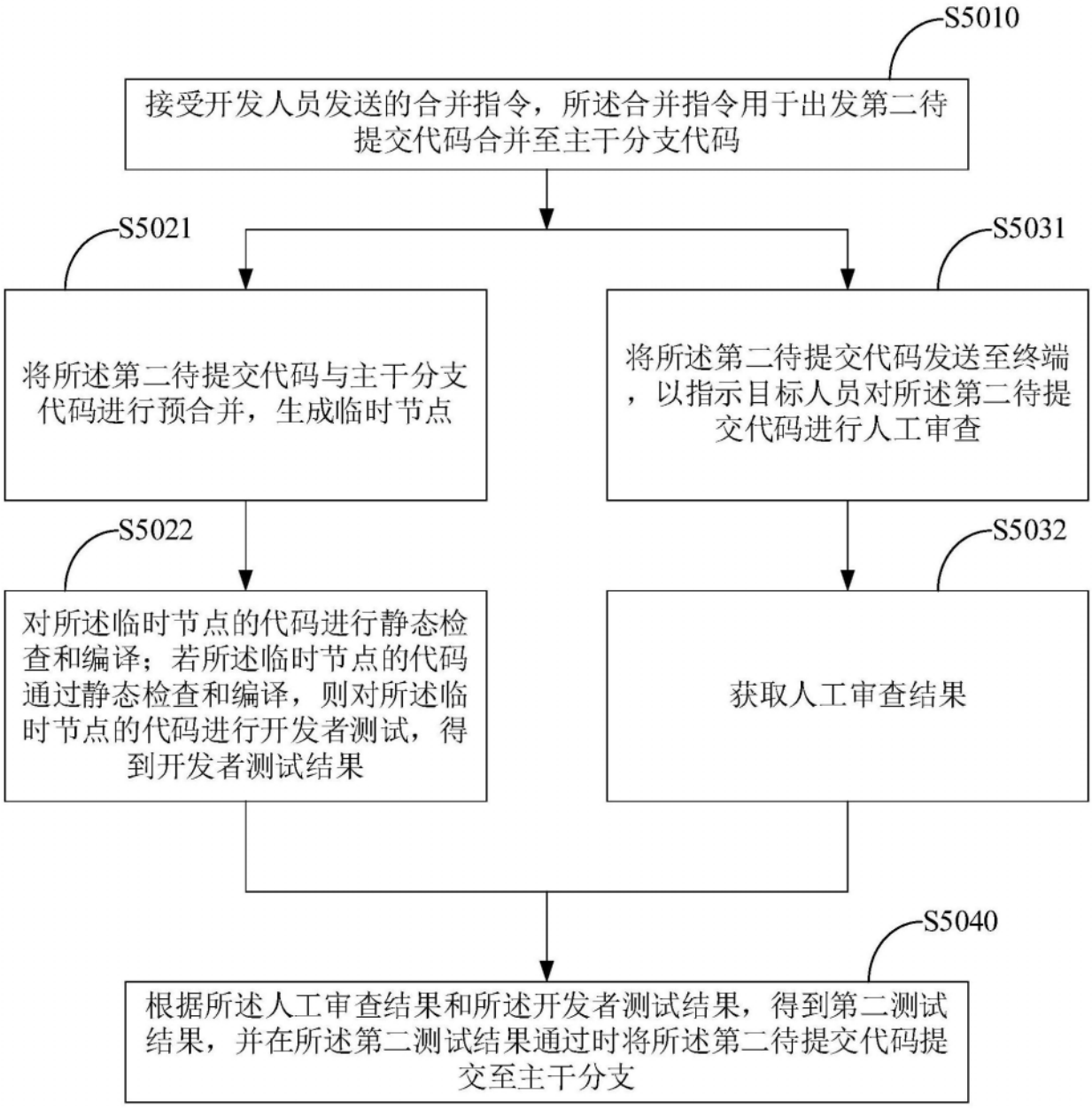

S5010
接受开发人员发送的合并指令，所述合并指令用于出发第二待提交代码合并至主干分支代码
S5021
将所述第二待提交代码与主干分支代码进行预合并，生成临时节点
S5031
将所述第二待提交代码发送至终端，以指示目标人员对所述第二待提交代码进行人工审查
S5022
对所述临时节点的代码进行静态检查和编译；若所述临时节点的代码通过静态检查和编译，则对所述临时节点的代码进行开发者测试，得到开发者测试结果
S5032
获取人工审查结果
S5040
根据所述人工审查结果和所述开发者测试结果，得到第二测试结果，并在所述第二测试结果通过时将所述第二待提交代码提交至主干分支

图5

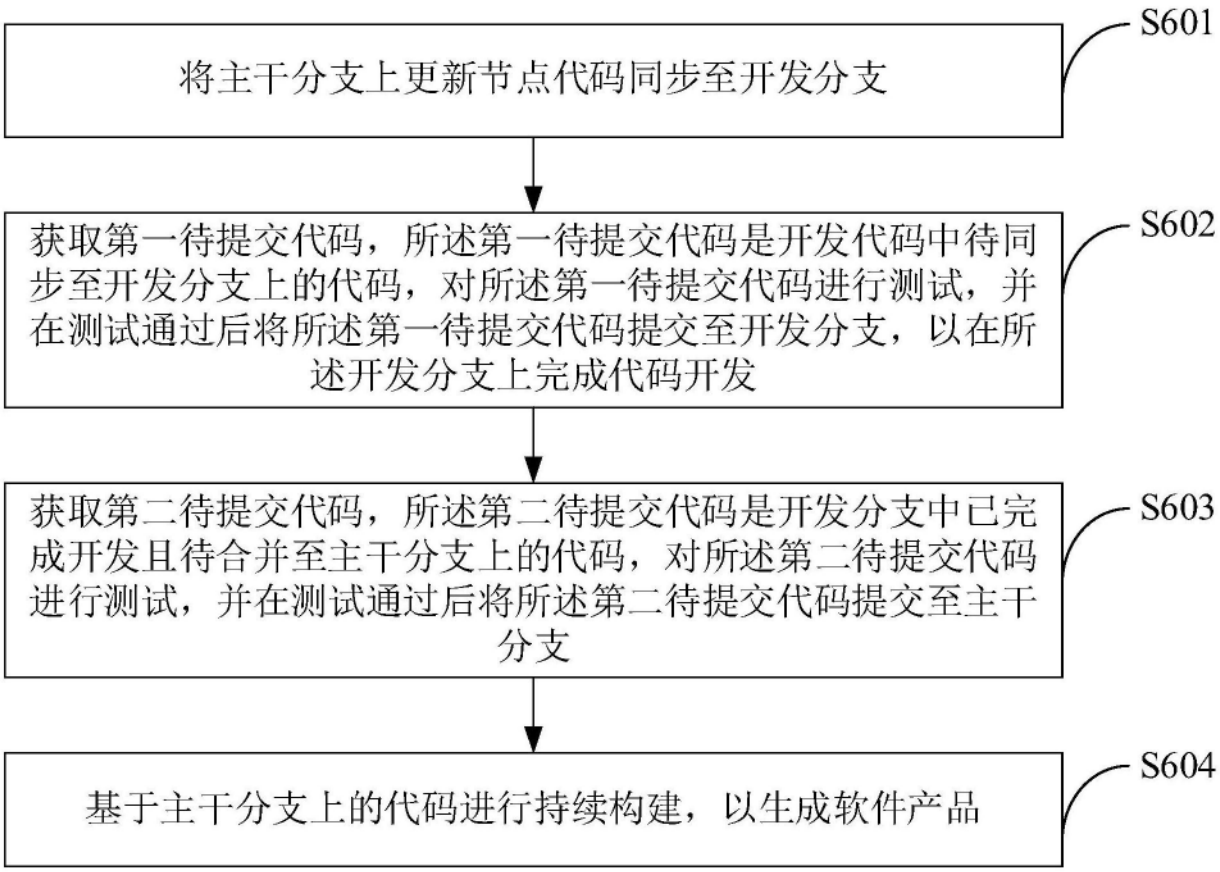

S601
将主干分支上更新节点代码同步至开发分支
S602
获取第一待提交代码，所述第一待提交代码是开发代码中待同步至开发分支上的代码，对所述第一待提交代码进行测试，并在测试通过后将所述第一待提交代码提交至开发分支，以在所述开发分支上完成代码开发
S603
获取第二待提交代码，所述第二待提交代码是开发分支中已完成开发且待合并至主干分支上的代码，对所述第二待提交代码进行测试，并在测试通过后将所述第二待提交代码提交至主干分支
S604
基于主干分支上的代码进行持续构建，以生成软件产品

图6

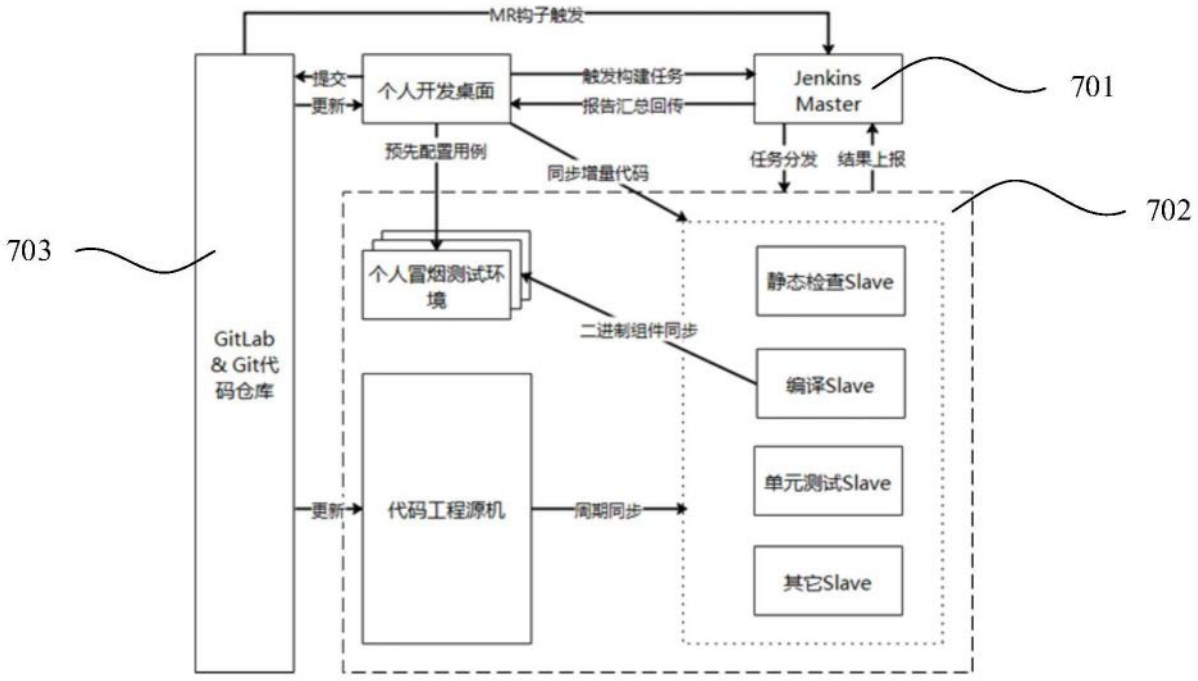

MR钩子触发
提交
更新
个人开发桌面
触发构建任务
报告汇总回传
Jenkins Master
701
预先配置用例
同步增量代码
任务分发
结果上报
702
703
个人冒烟测试环境
静态检查Slave
二进制组件同步
编译Slave
GitLab & Git代码仓库
代码工程源机
更新
周期同步
单元测试Slave
其它Slave

图7

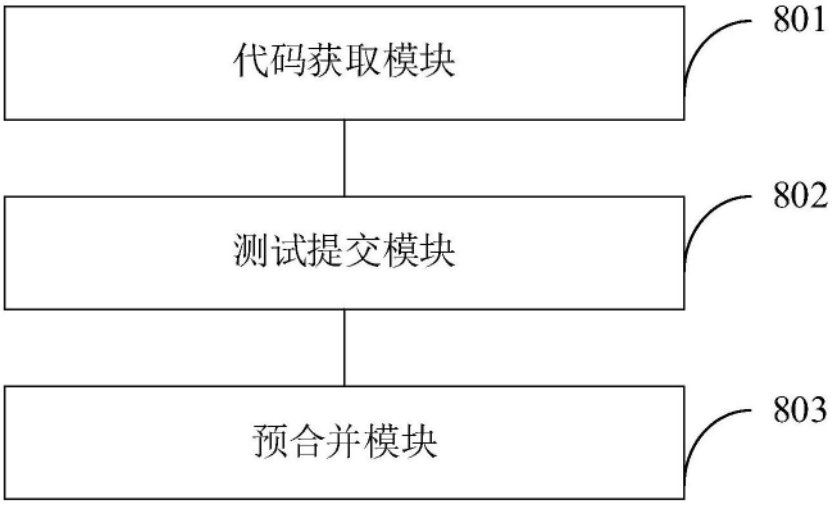


图8

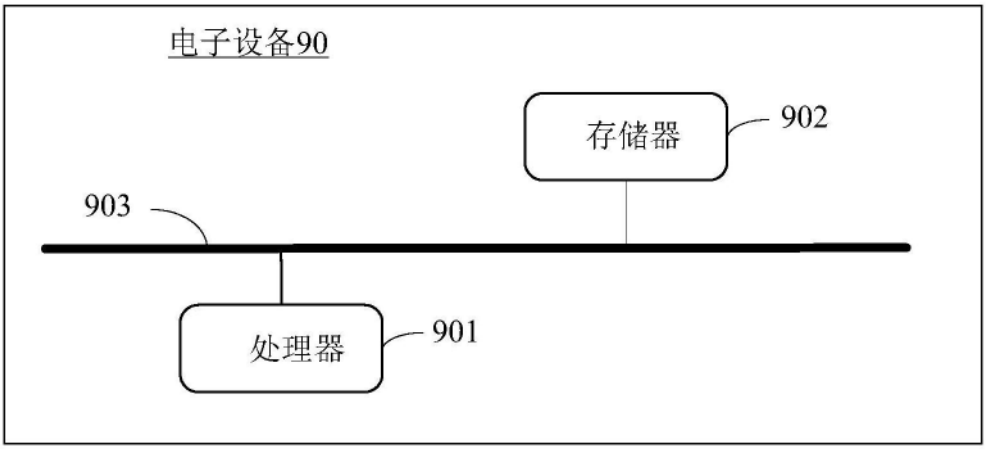


图9