



(12) 发明专利申请

(10) 申请公布号 CN 104246714 A

(43) 申请公布日 2014. 12. 24

(21) 申请号 201280072829. 1

(22) 申请日 2012. 04. 23

(85) PCT国际申请进入国家阶段日
2014. 10. 31

(86) PCT国际申请的申请数据
PCT/US2012/034676 2012. 04. 23

(87) PCT国际申请的公布数据
W02013/162503 EN 2013. 10. 31

(71) 申请人 惠普发展公司, 有限合伙企业
地址 美国德克萨斯州

(72) 发明人 D. 塔尔 I. 梅尔曼

(74) 专利代理机构 中国专利代理(香港)有限公
司 72001

代理人 谢攀 徐红燕

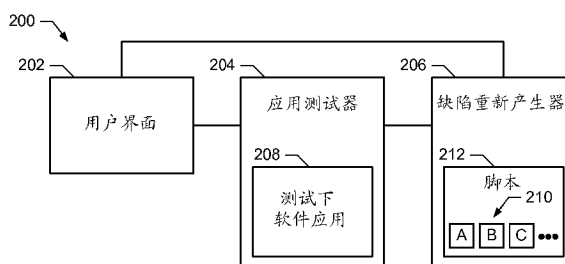
(51) Int. Cl.
G06F 11/36(2006. 01)
G06F 9/44(2006. 01)

权利要求书2页 说明书13页 附图16页

(54) 发明名称
软件缺陷验证

(57) 摘要

本发明公开了软件缺陷验证。示例性方法包括利用处理器访问表示要在执行要被测试的软件应用时被执行的一组动作的脚本, 该组动作与所报告缺陷相关联; 执行计算机上的要被测试的软件应用; 以及利用处理器经由要被测试的应用执行脚本中的该组动作以尝试重新产生所报告缺陷。



1. 一种方法,包括:

利用处理器访问表示要在执行要被测试的软件应用时被执行的一组动作的脚本,该组动作与所报告缺陷相关联;

执行计算机上的要被测试的软件应用;以及

利用所述处理器经由要被测试的应用执行所述脚本中的该组动作以尝试重新产生所述所报告缺陷。

2. 如权利要求 1 所述的方法,还包括:

将要被测试的软件应用中的缺陷报告为所报告缺陷;以及

将脚本附接至所报告缺陷。

3. 如权利要求 2 所述的方法,还包括:通过记录与所述软件应用的多个用户交互来生成所述脚本。

4. 如权利要求 1 所述的方法,还包括验证:当所述脚本中的该组动作已经被执行并且所述缺陷未被检测到时,已经从所述软件应用中移除了所述所报告缺陷。

5. 如权利要求 1 所述的方法,还包括:在执行所述脚本中的该组动作之前,将要被测试的软件应用置于初始状态。

6. 一种装置,包括:

应用测试器,用以执行测试下软件应用;以及

缺陷重新产生器,用以尝试通过在所述应用测试器执行所述测试下软件应用时执行在脚本中定义的一组动作来重新产生所报告缺陷,该组动作与所述所报告缺陷相关联。

7. 如权利要求 6 所述的装置,还包括:

脚本记录器,用以监视与所述测试下软件应用的用户交互;以及

脚本生成器,用以在用户界面接收到缺陷要被报告的指示时根据所监视的用户交互来生成脚本。

8. 如权利要求 7 所述的装置,其中所述脚本生成器生成所述脚本以包括继所述测试下软件应用的预定状态的发生之后发生的用户交互。

9. 如权利要求 6 所述的装置,还包括缺陷验证器,用以在该组动作已经被执行时确定是否已经从软件应用中移除了所报告缺陷。

10. 如权利要求 6 所述的装置,其中所述缺陷重新产生器在执行该组动作之前在所述测试下软件应用未处于初始状态时将所述测试下软件应用置于所述初始状态。

11. 一种计算机可读存储介质,包括计算机可读指令,所述计算机可读指令当由计算机执行时使得所述计算机至少进行以下操作:

访问表示要在执行要被测试的软件应用时被执行的一组动作的脚本,该组动作与所报告缺陷相关联;

执行计算机上的要被测试的软件应用;以及

经由要被测试的应用执行脚本中的该组动作以尝试重新产生所报告缺陷。

12. 如权利要求 11 所述的计算机可读存储介质,其中所述指令还使得所述计算机:

将要被测试的软件应用中的缺陷报告为所报告缺陷;以及

将脚本附接至所报告缺陷。

13. 如权利要求 11 所述的计算机可读存储介质,其中所述指令还使得所述计算机通过

记录与所述软件应用的多个用户交互来生成所述脚本。

14. 如权利要求 11 所述的计算机可读存储介质,其中所述指令还使得所述计算机验证:当所述脚本中的该组动作已经被执行并且所述缺陷未被检测到时,已经从所述软件应用中移除了所述所报告缺陷。

15. 如权利要求 11 所述的计算机可读存储介质,其中所述指令还使得所述计算机在执行所述脚本中的该组动作之前将要被测试的软件应用置于初始状态。

软件缺陷验证

背景技术

[0001] 软件测试是软件应用开发的通用部分。软件测试包括以可能预期终端用户与软件进行交互的方式与软件进行交互。当发现软件中的缺陷(或错误)时,对软件进行定义的底层计算机代码被修改以校正这些缺陷。

附图说明

[0002] 图 1 是根据本公开的教导而构造的示例性软件开发系统的框图。

[0003] 图 2 是可以被用来实现图 1 的软件测试器的示例性软件测试器的框图。

[0004] 图 3 是可以被用来实现图 1 的软件测试器的另一示例性软件测试器的框图。

[0005] 图 4A-4E 图示了根据本公开的教导的监视对软件应用进行测试期间的用户交互时的软件开发系统的示例性用户界面。

[0006] 图 5A-5D 图示了根据本公开的教导的验证所报告缺陷已经被修正时的软件开发系统的示例性用户界面。

[0007] 图 6 是表示当被执行时使得处理器报告测试下软件应用中的缺陷的示例性机器可读指令的流程图。

[0008] 图 7 是表示当被执行时使得处理器尝试重新产生测试下软件应用中的所报告缺陷的示例性机器可读指令的流程图。

[0009] 图 8 是表示当被执行时使得处理器尝试重新产生测试下软件应用中的所报告缺陷并且验证所报告缺陷是否继续存在于该软件应用中的示例性机器可读指令的流程图。

[0010] 图 9 是能够执行图 6-8 的指令以实现图 1、2 和 / 或 3 的软件测试器的示例性处理器平台的框图。

具体实施方式

[0011] 在现代软件开发中,软件应用开发者(本文也被简称为开发者)生成代码来实现软件应用。如软件开发中常见的那样,软件经常初始包含使得软件的行为偏离所期望或意图的行为的缺陷(或错误)。在开发者已经写出了所有或部分的软件代码之后,软件代码可以被发送至软件测试者(例如,质量保证工程师)以供测试。例如,软件测试者可以采用各种方式对软件进行测试以验证软件服从所意图的行为。

[0012] 当软件测试者识别出不服从所意图的行为的软件行为的实例时,软件测试者(例如,直接和 / 或间接地经由缺陷系统)将该缺陷报告至开发者。所报告缺陷范围可以从次要的(例如,装饰性的(cosmetic))到严重的(例如,核心功能问题)。开发者然后将计算机代码移除、修改、和 / 或添加至软件应用以修正所报告缺陷。当开发者修正了所报告缺陷时,开发者将软件的新版本提交至软件测试者。软件测试者然后验证所报告缺陷已经被修正。例如,软件测试者可以尝试重新产生先前导致了缺陷的发现的情况并确定是否仍然可以观察到缺陷。

[0013] 在过去,软件缺陷验证是手动过程。负责软件测试的人(例如,质量保证工程师)在

所报告缺陷已经被开发者(或者质量保证工程师可以是开发者自己)解决时被通知。该人然后必须通过手动地对软件应用重新测试并确定软件缺陷是否仍然存在来尝试重新产生该软件缺陷。手动测试因此会是昂贵且耗时的过程。进行手动测试所需的费用和时间可以导致针对软件的质量的妥协(例如,一些缺陷可能未被解决以便留在预算内等等)。

[0014] 示例性方法、装置和制品可以被用来在软件开发系统中执行软件缺陷验证。本文所公开的示例性方法、装置和制品通过使验证所报告的软件缺陷的过程自动化而克服了与现有技术相关联的问题。在一些示例中,自动化软件测试器记录用户进行软件应用的手动测试的动作,包括与测试下软件应用的交互。当用户报告软件应用中的缺陷时,软件测试器生成表示导致识别出缺陷的用户动作的脚本。软件测试器将脚本附接或附加到所报告缺陷。当所报告缺陷后来被该用户(或者不同用户)验证时,软件测试器执行该脚本以尝试重新产生该缺陷。开发者还可以使用脚本来可靠且快速地重新产生缺陷,这使得更多的高效解决方案能够报告缺陷。

[0015] 对比于摘要中的创建测试脚本的已知软件测试应用(即,不具有与软件缺陷的关联性),本文所公开的示例性方法、装置、和制品是缺陷中心式的(defect-centric)。示例性方法、装置、和制品将缺陷重新产生脚本附接或关联至所报告缺陷并且尝试响应于由用户对缺陷的选择而重新产生缺陷。以此方式,示例性方法、装置、和制品提供了对缺陷的快速验证和增强的软件开发效率。

[0016] 本文公开了示例性计算机可读指令,其使得处理器识别对要被测试的软件应用中的所报告缺陷的选择。基于该选择,指令使得处理器访问表示要由计算机在执行要被测试的软件应用时执行的一组动作的脚本。该组脚本与所选择的所报告缺陷相关联。示例性指令还使得处理器执行计算机上的要被测试的软件应用,并执行脚本中的该组动作以尝试重新产生所报告缺陷。

[0017] 本文所公开的示例性装置包括用户界面、应用测试器、和缺陷重新产生器。示例性用户界面接收针对要被测试的软件应用的所报告缺陷的选择。应用测试器执行测试下软件应用。示例性缺陷重新产生器尝试通过应用测试器执行测试下软件应用时执行脚本中所定义的一组动作来重新产生所选择的报告的软件缺陷。该组动作与所选择的报告的软件缺陷相关联。

[0018] 如本文所使用的,术语“验证缺陷”或“验证所报告缺陷”是指确定和/或确认已经根据某一准则满意地修正或解决了缺陷(例如,根据验证器的满意)。

[0019] 图1是示例性软件开发系统100的框图。图1的示例性系统100可以被用来执行针对开发和/或测试中的软件应用的软件缺陷验证。图1的示例包括应用开发器102、软件测试器104、缺陷管理器106、以及测试管理器108。

[0020] 图1的示例性应用开发器102被用来开发或生成软件应用。例如,应用开发器102可以是实现在一个或多个计算机、服务器、网络、和/或其他设备上的开发环境。诸如(一个或多个)软件工程师之类的一个或多个人使用应用开发器102来编写软件代码和/或生成要被测试的可执行软件。(一个或多个)软件应用可以被开发以尝试符合所接收到的软件应用要求110。例如,图1的软件应用要求110定义了软件应用的所期望的目标、目的、输入、输出、视觉要求、和/或行为。应用开发器102向软件测试器104提供用于测试的(一个或多个)软件应用。

[0021] 图 1 的示例性软件测试器 104 测试软件应用以识别、报告、和 / 或验证由应用开发器 102 提供的软件应用中(例如,软件应用的不同版本中)的缺陷。在一些示例中,软件测试器 104 是执行在计算机或处理平台(例如,图 9 的处理平台 900)上的测试工具。

[0022] 用户(例如,质量保证工程师)可以使用图 1 的示例性软件测试器 104 来对由应用开发器 102 提供的(一个或多个)软件应用 112 执行测试。当用户正在经由软件测试器 104 对软件应用 112 进行测试时,软件测试器 104 自动地记录用户和软件应用之间的交互(例如,经由键盘输入的数据、使用光标和 / 或鼠标选择的对象等等)。图 1 的软件测试器 104 将所记录的交互的表示存储在用户交互日志 114 中。

[0023] 当用户识别出测试下软件应用 112 中的缺陷时,用户经由软件测试器 104 报告缺陷。例如,用户可以经由软件测试器 104 生成缺陷记录。所报告缺陷可以包括例如缺陷标识符(例如,缺陷号)、所期望的行为、观察到的行为、可疑的原因、在其中观察到缺陷的状态或上下文、和 / 或用户相信在解决或修正所报告缺陷方面对于开发器有益的任何其他信息。

[0024] 响应于用户报告缺陷,示例性软件测试器 104 生成包括由用户进行的、导致了软件缺陷的交互(例如,日志 114 中记录的交互)的脚本。在一些示例中,软件测试器 104 将脚本附加或附接至所报告缺陷。此外或可替代地,软件测试器 104 将脚本存储在示例性测试管理器 108 中(例如,作为自动化测试)。在一些示例中,用户可以在将脚本附加至所报告缺陷之前手动地修改该脚本,以更精确地定义导致所报告缺陷的交互。

[0025] 图 1 的示例性缺陷管理器 106 从软件测试器 104 接收所报告缺陷并将所报告缺陷提供给应用开发器 102。在一些示例中,应用开发器从缺陷管理器 106 检索所报告缺陷(例如,在缺陷检查期间)。当应用开发器 102 解决(例如,修正)了从缺陷管理器 106 接收的所报告缺陷时,应用开发器返回或者更新缺陷管理器 106 中的所报告缺陷。示例性缺陷管理器 106 将已解决(但未验证)的缺陷提供给软件测试器 104 来进行验证。例如,软件测试器 104 可以在所报告缺陷验证时段期间访问缺陷管理器 106。

[0026] 示例性测试管理器 108 从软件测试器 104 接收(一个或多个)测试脚本 116。示例性测试脚本 116 包括执行一组步骤的缺陷重新产生指令 118,如果那些缺陷存在于软件应用中,那么该组步骤将产生一个或多个特定缺陷的证据。在一些示例中,测试脚本还包括验证指令 120,以识别一个或多个特定缺陷的证据,如果那些缺陷存在的话。当软件测试器 104 验证所报告缺陷已经被修正时,示例性软件测试器 104 访问来自测试管理器 108 的适当的测试脚本 116。在一些其他示例中,示例性软件测试器 104 访问附加至所报告缺陷(例如,在缺陷记录中)的脚本 116。

[0027] 示例性测试管理器 108 还可以维护一组脚本 116,以执行软件应用 116 的未来版本的自动化测试。例如,测试管理器 108 可以提供针对未来版本的自动化回归测试的测试以识别可能已经再次出现的任何缺陷。示例性软件测试器 104 将与已验证的缺陷相关联的脚本 116 提供给测试管理器 108,其包括未来自动化测试中的脚本 116。

[0028] 图 2 是可以被用来实现图 1 的软件测试器 104 的示例性软件测试器 200 的框图。图 2 的示例性软件测试器 200 包括用户界面 202、应用测试器 204、和缺陷重新产生器 206。软件测试器 200 可以被实现在例如下文参考图 9 所述的处理平台 900 上。

[0029] 图 2 的示例性用户界面 202 接收来自软件测试器 200 的用户的输入和 / 或将输出提供给软件测试器 200 的用户。例如,用户界面 202 可以包括以下中的一个或多个:用以向

用户图示虚拟显示的显示屏、用以接收数据输入(例如,击键、字母数字字符信息等等)的键盘、和/或用以控制光标和/或接收命令的鼠标。在软件测试和/或缺陷验证期间,示例性用户界面 202(例如,经由输入的组合)接收针对要被测试的软件应用的所报告的软件缺陷的选择。

[0030] 示例性应用测试器 204 执行测试下软件应用 208 并监视执行的软件应用。例如,应用测试器 204 可以接收要被测试的应用(例如,从图 1 的应用开发器 102)。示例性应用测试器 204 执行测试下软件应用 208,同时监视执行的应用的输入和输出(例如,网络连接、用户输入、对用户的输出、外围输入和/或输出、进入和/或出自应用的服务对象调用等等)。

[0031] 用户可以经由用户界面 202 与应用测试器 204 和/或要被测试的软件应用 208 进行交互。例如,用户界面 202 使得用户能够选择由应用测试器 204 提供的菜单项目或动作(例如,选择缺陷、验证缺陷、报告缺陷、执行应用 204 等等)。用户界面 202 还使得用户能够(经由应用测试器 204)与软件应用及其特征进行交互。

[0032] 图 2 的示例性缺陷重新产生器 206 尝试重新产生所选择的报告的软件缺陷(例如,经由用户界面 202 确定的)。为此,示例性缺陷重新产生器 206 在应用测试器 204 执行测试下软件应用 208 时自动地(例如,在不具有用户参与的情况下)执行脚本 212(例如,图 1 的脚本 116)中所定义的一组动作 210。图 2 的该组动作 210 包括用以重新产生所选择的报告的软件缺陷的指令(例如,如果缺陷未被修正的话)。在一些示例中,该组动作 210 在对软件应用 208(或者测试下软件应用 208 的早期版本)的在先测试期间被自动地记录并经由脚本 210 被附加至所报告缺陷。

[0033] 示例性软件测试器 200 的更详细操作在下面参考图 5A-5D、图 6、和/或图 7 来描述。

[0034] 图 3 是可以被用来实现图 1 的软件测试器 104 的另一示例性软件测试器 300 的框图。图 3 的示例包括图 2 的示例性用户界面 202、示例性应用测试器 204、和示例性缺陷重新产生器 206。然而,对比于图 2 的软件测试器 200,图 3 的示例性软件测试器 300 还包括缺陷验证器 314、脚本记录器 316、和脚本生成器 318。软件测试器 300 可以被实现在例如下文参考图 9 所述的处理平台 900 上。

[0035] 图 3 的示例性缺陷验证器 314 在一组动作 210 已经被执行时确定是否已经从软件应用 208 移除了所选择的报告的软件缺陷。例如,缺陷验证器 314 可以请求和/或接收来自用户界面 202 和/或应用测试器 204 的信息,该信息证明了所选择的报告的软件缺陷的存在或不存在。在一些示例中,脚本 212 包括用以获得和/或评估所报告缺陷是否已经被修正的一个或多个指令。缺陷验证器 314 使用这些指令来验证所报告缺陷。

[0036] 由缺陷验证器 314 获得和/或使用的信息和/或证据的类型可以依赖于特定的所报告缺陷而不同。例如,利用测试下软件应用 208 的用户界面的所报告缺陷(例如,不正确的图形)可以通过从用户界面 202 获得输出信息来验证。相比之下,与数据处理差错有关的所报告缺陷可以通过从应用测试器 204 获得测试下软件应用 208 的数据输入和/或输出来验证。

[0037] 图 3 的示例性脚本记录器 316 监视与测试下软件应用 208 的用户交互。例如,脚本记录器 316 从用户界面 202 接收输入。这些输入可以包括输入的类型(例如,鼠标点击、光标移动、光标位置、击键)、与输入相关联(例如,由输入所影响的)数据结构(例如,用户正

在键入字符到其中的数据字段、可点击的按钮等等)、所输入的数据(例如,字母数字字符等等)、和 / 或表征用户交互的任何其他信息。示例性脚本记录器 316 将所监视的交互存储在用户交互日志 320 中。在一些示例中,脚本记录器 316 自动地(即,在不具有用户参与的情况下)记录时间戳、到用户界面 202 的输出、到软件应用 208 的输入和 / 或输出(例如,经由应用测试器 204)、和 / 或对于重新产生软件应用中的缺陷有用或者必要的任何其他信息。

[0038] 基于在日志 320 中记录的交互,示例性脚本生成器 318 根据所监视的用户交互生成脚本(例如,脚本 212)。在图 3 的示例中,脚本生成器 318 在用户界面 202 接收到软件缺陷要被报告的指示时生成脚本。此类指示可以包括用户点击“报告缺陷”按钮(例如,当用户识别出测试下软件应用 208 中的缺陷时)。示例性脚本生成器 318 可以将所生成的脚本(例如,脚本 212)提供至缺陷重新产生器 206 和 / 或至缺陷验证器 314。在一些示例中,脚本生成器 318 将所生成的脚本 212 附加到缺陷的报告。

[0039] 可以被用来实现图 2 和 / 或 3 的应用测试器 204、脚本记录器 316、和 / 或脚本生成器 318 的示例性工具是由 Hewlett-Packard 开发的 QuickTest Professional™ 软件套件和 / 或也由 Hewlett-Packard 开发的 HP Functional Testing™ 软件。

[0040] 图 3 的示例性软件测试器 300 的更详细操作在下面参考图 4A-4E、图 5A-5D、图 6、图 7、和 / 或图 8 来描述。

[0041] 图 4A-4D 图示了软件开发系统 100 在对软件应用的测试期间监视用户交互时的软件开发系统(例如,图 1 的软件开发系统 100)的示例性用户界面 400。图 4A-4D 的示例性用户界面 400 可以在显示屏上被呈现给用户。

[0042] 所图示的示例的用户界面 400 包括与软件测试应用 402(例如,由图 2 和 / 或 3 的应用测试器 204)对应(例如,由其生成)的窗口。在图 4A-4D 的示例中,软件测试应用 402 正被用来测试正在生成由显示方框或窗口 404 实现的输出的软件应用(例如,图 2 的测试下软件应用 208)。所图示的示例的软件测试应用 402 提供了控制界面 403,用户能够利用所述控制界面 403 来发起软件应用的测试(例如,经由开始测试按钮 406)和 / 或结束软件应用的测试(例如,经由结束测试按钮 408)。

[0043] 在图 4A-4D 的示例中,示例性软件测试应用还使得用户能够将测试下软件应用的状态设为初始状态(例如,经由设置初始状态按钮 410)和 / 或使软件应用返回至初始状态(例如,经由返回至初始状态按钮 412)。如本文所使用的,术语“初始状态”是指测试要从其开始的开始点,其对应于测试下软件应用的任何状态(例如,模拟的系统状况,诸如存储器内容),从所述开始点起能够执行一组指令或步骤来尝试重新产生软件缺陷。例如,测试下软件应用的任何特定状态可以被指定为初始状态。然后软件应用的用户(例如,测试者)可以在测试期间、测试之后、和 / 或要开始测试的任何时间返回至初始状态。因此,脚本中的一组步骤可以使用初始状态(例如,所指定的开始点)来可靠地重新产生软件缺陷(如果软件缺陷还未被修正的话)和 / 或来验证缺陷已经被修正,这是因为用以重新产生软件缺陷的步骤从相同的系统和 / 或软件状况被发起,原因在于步骤初始地进行以发现缺陷。

[0044] 测试下软件应用的初始状态例如可以通过以下内容来规定:被分配给应用的存储器的内容、启用的(一个或多个)添加应用的(一个或多个)状态、软件应用的经编程的稳定状态(例如,主屏幕、菜单屏幕等等)、和 / 或规定了软件应用的状态的任何其他方法。在一些示例中,测试下软件应用 404 的状态可以与软件应用的后续版本的状态相兼容。

[0045] 图 4A-4D 的示例性软件测试应用还使得用户能够报告软件缺陷(例如,经由报告缺陷按钮 414)。当软件测试应用的用户识别出或观察到缺陷时,用户可以选择(例如,点击)报告缺陷按钮 414 来使得软件测试应用报告测试下软件应用中的缺陷。如下面更详细地描述的,报告软件缺陷可以包括生成规定了关于缺陷的信息的缺陷记录。

[0046] 图 4A 图示了软件测试应用正在测试测试下软件应用时的示例性用户界面 400。在图 4A 的示例中,测试下软件应用处于第一状态 416(例如,状态 0)。示例性第一状态 416 可以是当用户选择(例如,经由光标 418 点击)开始测试按钮 406 时测试下软件应用进入的第一状态。在选择开始测试按钮 406 时,用户和 / 或测试下软件应用的动作被监视(例如,经由图 2 和 / 或 3 的用户界面 202、应用测试器 204、和 / 或脚本记录器 316)。

[0047] 图 4B 图示了与测试下软件应用的示例性用户交互正在被监视时的图 4A 的用户界面 400。在图 4B 的示例中,测试下软件应用已经进入了另一状态 420(例如,状态 A)。

[0048] 在一些示例中,用户可以通过经由用户界面 400 和 / 或软件测试应用与测试下软件应用进行交互来使得测试下软件应用进入状态 420。当测试下软件应用进入状态 420 时,用户可以将状态 420 设置(或分配)为初始状态(例如,通过利用光标选择设置初始状态按钮 410)。

[0049] 在一些其他示例中,用户可以通过经由光标 418 选择返回至初始状态按钮 412 来使得测试下软件应用返回至初始状态(例如,状态 420)。当测试下软件应用处于初始状态 420 时,图 3 的示例性脚本记录器 316 记录测试下软件应用处于初始状态。该信息随后被用来向脚本提供一组动作要从其被执行的状态(例如,初始状态 420)。

[0050] 图 4C 图示了继初始状态之后的示例性用户交互正在被监视时的图 4A 的用户界面 400。在图 4C 的示例中,用户通过在位置 B 处点击(例如,经由鼠标)由测试下软件应用(例如,在应用窗口中)输出的显示方框内的某一位置处的光标 418 来进行测试下软件应用的测试。图 3 的示例性日志记录器 316 记录了经由示例性用户界面 400(例如,图 3 的用户界面 202)接收到的用户交互。

[0051] 示例性测试随着用户将光标 418 移动 422 至由测试下软件应用输出的显示方框 404 内的第二位置 C 以及通过将光标 418 移动 424 至由测试下软件应用输出的显示方框 404 内的另一位置 D 而继续。示例性日志记录器 316 可以将经由用户界面 400 接收到的这些用户交互记录为单独的动作和 / 或为单个动作(例如,基于移动 422 是否修改了测试下软件应用的状态)。

[0052] 用户然后选择(例如,经由鼠标点击、“tab”击键等等)由测试下软件应用输出的显示方框 404 内的数据输入字段(例如,文本框 426)并通过做出两次击键(例如,“1”击键跟着“5”击键)来输入数字(例如,“15”)。示例性日志记录器 316 记录经由用户界面 400 的这些用户交互。与移动交互 422、424 一样,示例性日志记录器 316 可以将这些用户交互记录为单独的动作和 / 或单个动作。

[0053] 例如,如果“1”击键或“5”击键(或者这二者都)导致软件状态的改变(例如,所分配的存储器内容的改变),那么(一个或多个)击键可以被视为单独的动作。

[0054] 数据输入按钮(例如,“去往”)428 然后被选择(例如,通过利用鼠标点击、通过敲击键盘上的“回车”键等等)。根据所意图的软件行为,在用户选择了“去往”按钮 428 之后,测试下软件应用应当处于状态 E。示例性日志记录器 316 记录按钮 428 的选择以与用户选择

按钮 428 的方法(例如,鼠标点击、击键等等)一致。然而,示例性日志记录器 316 还可以记录该动作的等同物(例如,除了将光标 418 移动到按钮上并点击鼠标之外,记录按下“回车”键)。

[0055] 图 4D 图示了当要由用户报告示例性软件缺陷时的图 4A 的用户界面 400。根据示例性测试下软件应用的预期行为,数据输入字段 426 中显示的值要被改变为另一数字(例如,“16”)。然而,在所图示的示例中,字段 426 的值保持为所输入的数字“15”。此外,如根据示例性软件要求(例如,图 1 的软件应用要求 110)预期的,测试下软件应用保持处于状态 D 而不是转变为状态 E。因而,用户识别出存在软件缺陷并经由光标 418 选择报告缺陷按钮 414。

[0056] 图 4E 图示了当软件测试应用呈现对话框 430 以供用户报告测试下软件应用中的缺陷时的图 4A 的用户界面 400。示例性软件测试应用响应于用户选择图 4A-4D 的报告缺陷按钮 414 而呈现对话框 430。

[0057] 在图 4E 的示例中,对话框 430 呈现了缺陷记录 432(例如,由图 2 和 / 或 3 的应用测试器 204 生成的)。缺陷记录 432 包括信息,包括:报告用户的标识 434、缺陷标识符 436、缺陷日期(例如,时间戳)438、缺陷严重程度 440、测试下软件应用的版本的标识 442、以及缺陷重新产生脚本 444。在一些示例中,用户可以将评论、图示了缺陷的屏幕截图、和 / 或其他信息附加至缺陷记录 432。

[0058] 示例性缺陷重新产生脚本 444(例如,图 1 的脚本 116、图 2 和 / 或 3 的脚本 212)包括表示可以被执行以尝试重新产生在缺陷记录 432 中所报告缺陷的一组动作(例如,图 2 和 / 或 3 的动作 210)的指令 446。在图 4E 的示例中,缺陷重新产生脚本 444 由脚本生成器(例如,图 3 的脚本生成器 318)初始地填充有指令 446。用户可以添加、删除、和 / 或修改指令 446。例如,脚本 444 包括由用户添加的验证指令 448。示例性验证指令 448 使得缺陷验证器(例如,图 3 的缺陷验证器 314)能够验证缺陷已经被修正。

[0059] 当缺陷重新产生脚本 444 被满意地完成时,用户可以选择附加脚本按钮 450(例如,经由光标 418)来使得脚本生成器 318 将脚本 444 附加至记录 432。在一些其他示例中,脚本 444 可以被自动地附加至缺陷记录 432 和 / 或可以集成到缺陷记录 432。当缺陷记录 432 已经由用户满意地(例如,针对用户的满意)准备好时,用户可以选择输入缺陷按钮 452。

[0060] 尽管图 4A-4E 图示了测试软件应用、记录用户交互、以及报告缺陷的示例,但示例可以被修改以测试(一个或多个)任何类型的软件应用并记录(一个或多个)任何类型的交互。所记录的交互、所执行的测试、和 / 或所报告缺陷基于软件应用要求(例如,所意图的行为)和软件应用的编码。因而,所监视的交互和所报告缺陷的几乎无限制数量的变形是可能的。图 3 的示例性软件测试器 300 可以被用来报告此类缺陷并生成表示导致识别出或观察到缺陷的交互的脚本。

[0061] 图 5A-5D 图示了验证所报告缺陷已经被修正时的软件开发系统(例如,图 1 的软件开发系统 100)的示例性用户界面 500。图 5A-5D 的示例性用户界面 500 可以与图 4 的用户界面 400 相同或类似和 / 或可以被用来实现图 2 和 / 或 3 的用户界面 202。

[0062] 图 5A-5D 的示例性用户界面 500 包括由软件测试应用所生成的显示方框或窗口 502(例如,用户能够利用其与软件测试应用进行交互的用户界面 500 上的显示窗口)。图 5A-5D 的软件测试应用可以实现图 1-3 的示例性软件测试器 104、200、300 中的任何和 / 或

所有。在图 5A-5D 的示例中操作的示例性软件测试应用可以与图 4A-4E 的软件测试应用相同或不同。

[0063] 此外,在图 5A-5D 的示例中操作的软件测试应用可以在与图 4A-4E 的软件测试应用相同或不同的处理平台上被执行。因而,所报告缺陷可以在与缺陷被报告的相同的处理平台和 / 或软件测试应用上被验证。可替代地,所报告缺陷可以在与缺陷被报告的不同的处理平台和 / 或软件测试应用上被验证。有利地,即使未经由软件测试应用报告缺陷,图 5A-5D 的软件测试应用也使用脚本来高效且快速地验证软件缺陷。

[0064] 图 5A-5D 的示例性软件测试应用要验证测试下软件应用。在图 5A-5D 的示例中,示例性测试下软件应用是图 4A-4E 的测试下软件应用的后续版本。例如,测试下软件应用已经被修改(例如,由软件开发器)以尝试解决或修正缺陷。

[0065] 图 5A 图示了用户正在选择要被验证的所报告缺陷时的示例性用户界面 500。示例性用户界面 500 显示了与测试下软件应用对应的显示窗口 504(例如,由测试下软件应用生成的)。显示窗口 504 可以被显示为与由软件测试应用所生成的窗口 502 并排和 / 或被显示为窗口 502 的子窗口。

[0066] 在图 5A 的所图示的示例中,软件测试应用包括选择工具(诸如,填充有针对软件应用的所报告缺陷的下拉框 506)。用户可以选择下拉框 506,并且然后从所列出的缺陷中选择所报告缺陷 508(例如,“缺陷 F347”)。示例性所选择的所报告缺陷 F347 是与图 4E 中图示的示例性所报告缺陷相同的所报告缺陷。因此,所选择的所报告缺陷与用以尝试重新产生所报告缺陷的脚本相关联(例如,包括、附加有)。

[0067] 图 5B 图示了用户正在选择以验证所选择的所报告缺陷时的图 5A 的示例性用户界面 500。在图 5B 的示例中,用户已经选择了验证缺陷按钮 510。响应于接收到针对所选择的所报告缺陷 508 的验证缺陷按钮 510 的选择,示例性软件测试应用尝试通过执行脚本(例如,被包括有和 / 或附加于针对所报告缺陷的缺陷记录的脚本、图 1 的脚本 116、图 2 和 / 或 3 的脚本 212)来重新产生所报告缺陷。用户可以使用例如光标 512 来选择验证缺陷按钮 510。

[0068] 响应于用户选择验证所报告缺陷 508,示例性软件测试应用开始执行脚本。在所图示的示例中,软件测试应用首先基于脚本将测试下软件应用置于第一状态 A(例如,图 4B 的初始状态 420)。

[0069] 图 5C 图示了当软件测试应用已经执行了附加至所选择的所报告缺陷的脚本时的图 5A 的示例性用户界面 500。根据示例性脚本,测试下软件应用应当处于状态 D,其中击键“15”已经被输入到字段 514 中并且“去往”按钮 516 已经被选择(例如,经由图 2 和 / 或 3 的缺陷重新产生器 206 和用户界面 202)。在所图示的示例中,软件测试应用还示出了所报告缺陷的位置(例如,应当观察到缺陷的地方)和 / 或所预期的内容的注释 518 突显。

[0070] 在图 5C 的示例中,示例性测试下软件应用已经执行了计算并使字段 514 填充有值“16”。该值与突显 518 中示出的所预期值一致。因而,用户可以通过选择验证修正按钮 520 来确认或验证所报告缺陷已经被修正。

[0071] 图 5D 图示了当软件测试应用已经执行了附加至所选择的所报告缺陷的脚本并且所报告缺陷继续存在时的图 5A 的示例性用户界面 500。如图 5D 中所示,示例性字段 514 中的值不匹配于注释 518 中图示的值。当用户识别出缺陷继续存在时(例如,通过观察注释

518 和字段 514 之间的差异),用户可以选择拒绝修正按钮 522 来拒绝修正(例如,重新打开或将所报告缺陷返回至软件开发器以被解决)。

[0072] 尽管实现图 1 的软件测试器 104 的示例性方式已经在图 2 和 3 中进行了说明,但图 2 和 / 或 3 中所图示的元件、处理器和 / 或设备中的一个或多个可以被组合、分开、重新布置、省略、消除、和 / 或以任何其他方式实现。此外,示例性用户界面 202、示例性应用测试器 204、示例性缺陷重新产生器 206、示例性测试下软件应用 208、示例性一组动作 210、示例性脚本 212、示例性缺陷验证器 314、示例性脚本记录器 316、示例性脚本生成器 318、和 / 或更一般地图 1-3 的示例性软件测试器 104、200、300 可以由硬件、软件、固件和 / 或硬件软件和 / 或固件的任何组合来实现。因此,例如,示例性用户界面 202、示例性应用测试器 204、示例性缺陷重新产生器 206、示例性测试下软件应用 208、示例性一组动作 210、示例性脚本 212、示例性缺陷验证器 314、示例性脚本记录器 316、示例性脚本生成器 318、和 / 或更一般地图 1-3 的示例性软件测试器 104、200、300 中的任何一个可以由(一个或多个)电路、(一个或多个)可编程处理器、(一个或多个)专用集成电路(ASIC)、(一个或多个)可编程逻辑设备(PLD)、和 / 或(一个或多个)现场可编程逻辑设备(FPLD)等等来实现。当本专利的任何装置或系统权利要求被读以涵盖纯软件和 / 或固件实现时,示例性用户界面 202、示例性应用测试器 204、示例性缺陷重新产生器 206、示例性测试下软件应用 208、示例性一组动作 210、示例性脚本 212、示例性缺陷验证器 314、示例性脚本记录器 316、和 / 或示例性脚本生成器 318 中的至少一个据此被明确地定义为包括诸如存储器、DVD、CD、蓝光等等存储软件和 / 或固件的有形计算机可读介质。此外,图 1-3 的示例性软件测试器 104、200、300 仍可以包括除在图 4 中图示的那些之外或代替在图 4 中图示的那些的元件、过程和 / 或设备,和 / 或可以包括所图示的元件、过程和设备中任何一个或上所有的多于一个。

[0073] 表示用于实现图 1-3 的软件测试器 104、200、300 中任何一个的示例性机器可读指令的流程图被示出在图 6-8 中。在此示例中,机器可读指令包括供处理器执行的程序,诸如下面结合图 9 所讨论的示例性计算机 900 中示出的处理器 912。程序可以被体现在存储在诸如计算机可读存储介质(例如,CD-ROM、软盘、硬盘、数字多功能盘(DVD)、蓝光盘、或者与处理器 912 相关联的存储器)之类的有形计算机可读介质上的软件中,但整个程序和 / 或其部分能够替代地由不同于处理器 912 的设备执行和 / 或被体现在固件或专用硬件中。此外,尽管示例性程序参考图 6-8 中图示的流程图来描述,但可以替代地使用实现示例性软件测试器 104、200、300 的许多其他方法。例如,可以改变块的执行次序,和 / 或可以改变、消除、或者组合所描述的块中的一些。

[0074] 如上所述,图 6-8 的示例性过程可以使用存储在有形计算机可读介质上的经编码指令(例如,计算机可读指令)来实现,所述有形计算机可读介质诸如硬盘驱动、闪存、只读存储器(ROM)、光盘(CD)、数字多功能盘(DVD)、高速缓存、随机存取存储器(RAM)、和 / 或信息在任何持续时间(例如,在扩展的时间段、永久地、简短实例、用于暂时缓冲、和 / 或用于信息的高速缓存)内被存储在其中的任何其他存储介质。如本文所使用的,术语“有形计算机可读介质”被明确地定义为包括任何类型的计算机可读存储并排除传播信号。此外或替代地,图 6-8 的示例性过程可以使用存储在非临时计算机可读介质上的经编码指令(例如,计算机可读指令)来实现,所述非临时计算机可读介质诸如硬盘驱动、闪存、只读存储器、光盘、数字多功能盘、高速缓存、随机存取存储器、和 / 或信息在任何持续时间(例如,在扩展

的时间段、永久地、简短实例、用于暂时缓冲、和 / 或用于信息的高速缓存) 内被存储在其中的任何其他存储介质。如本文所使用的, 术语“非临时计算机可读介质”被明确地定义为包括任何类型的计算机可读介质并排除传播信号。如本文所使用的, 当短语“至少”被用作权利要求的前序中的过渡用语时, 它以与术语“包括”是开放式的相同方式是开放式的。因此, 使用“至少”作为其前序中的过渡用语的权利要求可以包括除了权利要求中明确记载的那些之外的元素。

[0075] 图 6 是表示示例性机器可读指令 600 的流程图, 所述示例性机器可读指令当被执行时使得处理器报告测试下软件应用中的缺陷。示例性指令 600 可以由图 9 的示例性处理器平台 900 执行以实现图 3 的用户界面 202、应用测试器 204、脚本记录器 316、和 / 或脚本生成器 318。

[0076] 示例性指令 600 开始于监视(例如, 经由图 3 的脚本记录器 316)经由用户界面(例如, 图 3 的用户界面 302、图 4 的用户界面 400)的用户交互(块 602)。例如, 脚本记录器 316 可以监视用户界面 202 以识别诸如击键、鼠标移动、鼠标点击、音频输入、成像设备输入、和 / 或任何其他类型的用户交互之类的用户输入。如果用户交互已经被识别(块 604), 则示例性脚本记录器 316 将用户交互记录在用户交互日志(例如, 日志 320)中(块 606)。

[0077] 当用户交互已经被记录(块 606)时, 或者在用户交互还未被识别(块 604)的情况下, 示例性应用测试器 204 确定缺陷是否已经被识别(例如, 用户是否已经指示了已经经由用户界面 202 观察到缺陷)(块 608)。如果缺陷还未被识别(块 608), 则控制返回至块 602 以继续监视用户交互。

[0078] 如果缺陷已经被识别(块 608), 则示例性应用测试器 204 生成缺陷记录(例如, 图 4E 的缺陷记录 432)(块 610)。示例性缺陷记录 432 可以包括所分配的缺陷标识符、生成了缺陷记录的用户的标识符、来自用户的附注、时间戳、缺陷的严重程度、测试下软件应用(例如, 图 2 和 3 的测试下软件应用 208 和 / 或图 4 的测试下软件应用)的版本。

[0079] 图 3 的示例性脚本生成器 318 根据用户交互日志 320 生成脚本(例如, 图 3 的脚本 212、图 4E 的缺陷重新产生脚本 444)(块 612)。脚本 444 包括针对缺陷重新产生器(例如, 图 2 和 3 的缺陷重新产生器 206)的表示用户交互的指令, 所述用户交互诸如例如是自最近发生的指定初始状态(例如, 图 4B 的状态 A 120)起的用户交互、自测试开始起(例如, 自用户选择了图 4A-4E 的开始测试按钮 406 起)的用户交互; 指定数目的最近交互(例如, 最后 30 次交互)、和 / 或交互的任何其他数目或表示。

[0080] 示例性脚本生成器 318 将脚本 444 附接(例如, 附加)至缺陷(例如, 至缺陷记录 432)(块 614)。例如, 脚本生成器 318 可以自动地将脚本 444 包括(例如, 附加)在缺陷记录中。示例性应用测试器 204 报告测试下软件应用中的缺陷(块 616)。例如, 应用测试器 204 可以将包括脚本 444 的缺陷记录 432 提供至缺陷管理器 106 和 / 或至应用开发器 102。

[0081] 在报告了缺陷(块 616)之后, 图 6 的示例性指令 600 可以结束。示例性指令 600 然后可以在用户开始另一测试时从开始重新开始。在一些其他示例中, 指令 600 可以在不具有附加的用户命令的情况下(例如, 在不具有重新开始指令 600 的情况下)使控制返回至块 602 以继续监视用户交互。

[0082] 图 7 是表示示例性机器可读指令 700 的流程图, 所述示例性机器可读指令 700 当被执行时使得处理器尝试重新产生测试下软件应用中的所报告缺陷。示例性指令 700 可以

由图 9 的示例性处理器平台 900 执行以实现图 1、2 和 / 或 3 的示例性软件测试器 104、200、300。

[0083] 图 7 的指令 700 开始于识别(例如,经由图 2 和 3 的用户界面 202)针对要被测试的软件应用的所报告的软件缺陷的选择(块 702)。例如,用户界面 202 可以从用户接收指示了针对测试下软件应用(例如,图 5A-5D 的测试下软件应用)的所报告的软件缺陷(例如,图 5B-5D 的软件缺陷 508)的选择的一个或多个命令。

[0084] 缺陷重新产生器(例如,图 2 和 / 或 3 的示例性缺陷重新产生器 206)访问表示一组动作的脚本以重新产生所选择的所报告缺陷 508(块 704)。例如,缺陷重新产生器 206 可以接收被包括在缺陷记录(例如,图 4E 的缺陷记录 432)中的脚本(例如,图 4E 的缺陷重新产生脚本 444)。在一些示例中,缺陷重新产生器 206 访问来自缺陷管理器(例如,图 1 的缺陷管理器 106)的缺陷记录 432 和 / 或脚本 444。

[0085] 应用测试器(例如,图 2 和 / 或 3 的示例性应用测试器 204)执行要被测试的软件应用(例如,图 5A-5D 的测试下软件应用 504)(块 706)。缺陷重新产生器 206 执行该组动作(例如,脚本 444 中的动作)以尝试重新产生所报告缺陷(块 708)。通过执行该组动作,示例性缺陷重新产生器 206 尝试重新产生用户(例如,质量保证工程师)先前经由其确定所选择的缺陷存在的状况。如果缺陷未被修正,则选择了所报告缺陷以供验证的用户可以观察到缺陷仍存在而不必手动地回溯所述步骤。另一方面,如果缺陷已经被修正,则用户可以观察到所报告缺陷已经被修正而不必手动地回溯所述步骤。

[0086] 示例性指令 700 然后可以结束。在一些示例中,指令 700 可以使控制返回至块 702 以识别针对验证的另一个所报告缺陷的选择。以此方式,质量保证工程师可以快速验证多个所报告缺陷已经被修正。

[0087] 图 8 是表示示例性机器可读指令 800 的流程图,所述示例性机器可读指令 800 当被执行时使得处理器尝试重新产生测试下软件应用中的所报告缺陷并且验证所报告缺陷是否继续存在于软件应用中。示例性指令可以由图 9 的示例性处理器平台 900 来执行以实现图 1 和 / 或 3 的示例性软件测试器 104、300。

[0088] 图 8 的示例性指令 800 开始于识别(例如,经由图 3 的用户界面 202)针对要被测试的软件应用的所报告的软件缺陷的选择(块 802)。例如,用户界面 202 可以从用户接收指示了针对测试下软件应用(例如,图 5A-5D 的测试下软件应用 504)的所报告的软件缺陷(例如,图 5B-5D 的软件缺陷 508)的选择的一个或多个命令。

[0089] 缺陷重新产生器(例如,图 3 的示例性缺陷重新产生器 206)访问表示一组动作的脚本以重新产生所选择的所报告缺陷 508(块 804)。例如,缺陷重新产生器 206 可以接收被包括在缺陷记录(例如,图 4E 的缺陷记录 432)中的脚本(例如,图 4E 的缺陷重新产生脚本 444)。在一些示例中,缺陷重新产生器 206 访问来自缺陷管理器(例如,图 1 的缺陷管理器 106)的缺陷记录 432 和 / 或脚本 444。

[0090] 应用测试器(例如,图 3 的示例性应用测试器 204)执行要被测试的软件应用(例如,图 5A-5D 的测试下软件应用)(块 806)。示例性缺陷重新产生器 206 确定测试下软件应用是否处于重新产生所报告缺陷的正确状态(块 808)。例如,缺陷重新产生器 206 可以识别用于执行一组动作的初始状态(例如,图 4B 和 / 或 5B 的状态 A)。

[0091] 初始状态可以根据脚本 444、由用户所规定的针对测试下软件应用的缺省初始状

态、和 / 或初始状态信息的任何其他源来确定。例如,缺陷重新产生器 206 可以通过以下内容来确定测试下软件应用是否处于初始状态:将被分配给测试下软件应用的存储器的内容与针对初始状态的存储器的内容进行比较、确定任何启用的添加应用的状态、确定测试下软件应用是否处于已编程的稳态(例如,在主屏幕上、在所指定的菜单屏幕上等等)和 / 或以其他方式将测试下软件应用的状态与初始状态进行比较。

[0092] 如果测试下软件应用未处于正确的(例如,初始的)状态(块 808),则示例性缺陷重新产生器 206 将测试下软件应用置于正确(例如,初始的)状态(块 810)。例如,缺陷重新产生器 206 可以经由图 3 的用户界面 202 执行与测试下软件应用的一个或多个交互。在一些其他示例中,缺陷重新产生器 206 向应用测试器 204 提供要被置于测试下软件应用 504 的所分配的存储器中的数据,以将测试下软件应用置于初始状态。

[0093] 在将测试下软件应用置于初始状态(块 810)之后,或者在缺陷重新产生器 206 确定测试下软件应用处于初始状态(块 808)的情况下,示例性缺陷重新产生器 206 执行一组动作(例如,脚本 444 中的动作)以尝试重新产生所报告缺陷(块 812)。通过执行该组动作,示例性缺陷重新产生器 206 尝试重新产生用户(例如,质量保证工程师)先前经由其确定所选择的缺陷存在的状况。

[0094] 缺陷验证器(例如,图 3 的缺陷验证器 314)确定所报告缺陷是否仍然存在(块 814)。例如,缺陷验证器 314 可以分析脚本 444 以确定是否存在任何验证指令。在一些其他示例中,缺陷验证器 314 针对指示了缺陷的存在或不存在的用户交互(例如,用户选择图 5C 和 5D 的“验证修正”按钮 520 或“拒绝修正”按钮 522)监视用户界面 202。

[0095] 如果缺陷验证器 314 确定缺陷不存在(块 814),则示例性缺陷验证器 314 将所报告缺陷标记为已修正(块 816)。例如,缺陷验证器 816 可以改变缺陷记录 432 中的状态或其他信息。另一方面,如果缺陷仍然存在(块 814),则缺陷验证器 816 返回所报告缺陷(例如,至图 1 的应用开发器 102 和 / 或至缺陷管理器 106)(块 818)。

[0096] 示例性指令 800 然后可以结束。在一些示例中,指令 800 可以使控制返回至块 802 以识别针对验证的另一所报告缺陷的选择。

[0097] 图 9 是能够执行图 6-8 的指令 600、700、800 以实现图 1-3 的软件测试器 102、200、和 / 或 300 的示例性处理器平台 900 的框图。计算机 900 例如能够是服务器、个人计算机、或者任何其他类型的计算设备。

[0098] 即时示例的处理器平台 900 包括处理器 912。例如,处理器 912 能够由来自任何期望的家族或制造商的一个或多个微处理器或控制器来实现。图 9 的示例性处理器 912 实现了图 3 的软件测试器 300,包括示例性应用测试器 204、示例性缺陷重新产生器 206、示例性测试下软件应用 208、示例性缺陷验证器 314、示例性脚本记录器 316、和 / 或示例性脚本生成器 318。

[0099] 处理器 912 包括本地存储器 913(例如,高速缓存)并且经由总线 918 与包括易失性存储器 914 和非易失性存储器 916 的主存储器进行通信。易失性存储器 914 可以由同步动态随机存取存储器(SDRAM)、动态随机存取存储器(DRAM)、RAMBUS 动态随机存取存储器(RDRAM)和 / 或任何其他类型的随机存取存储器设备来实现。非易失性存储器 916 可以由闪存和 / 或任何其他期望类型的存储器设备来实现。对主存储器 914、916 的访问由存储器控制器来控制。示例性本地存储器 913、示例性易失性存储器 914、和 / 或示例性非易失性

存储器 916 中的任何一个可以存储表示测试下软件应用 208、脚本 212、和 / 或用户交互日志 320 的数据和 / 或指令。示例性应用测试器 204、示例性缺陷重新产生器 206、示例性测试下软件应用 208、示例性缺陷验证器 314、示例性脚本记录器 316、和 / 或示例性脚本生成器 318、和 / 或更一般地示例性处理器 912 访问来自本地存储器 913、易失性存储器 914、和 / 或非易失性存储器 916 中的任何一个的测试下软件应用 208、脚本 212、和 / 或用户交互日志 320。

[0100] 处理器平台 900 还包括接口电路 920。接口电路 920 可以由任何类型的接口标准来实现,诸如以太网接口、通用串行总线(USB)、和 / 或 PCI 高速接口。

[0101] 一个或多个输入设备 922 被连接至接口电路 920。(一个或多个)输入设备 922 准许用户将数据和命令输入到处理器 912 中。(一个或多个)输入设备能够由例如键盘、鼠标、触摸屏、轨迹板、轨迹球、isopoint、和 / 或语音识别系统来实现。

[0102] 一个或多个输出设备 924 也连接至接口电路 920。输出设备 924 能够例如由显示设备(例如,液晶显示器、阴极射线管显示器(CRT)、打印机、和 / 或扬声器)来实现。因此,接口电路 920 通常包括图形驱动卡。示例性接口电路 920、(一个或多个)示例性输入设备 922、和 / 或(一个或多个)示例性输出设备 924 可以被结合使用以实现图 2 和 / 或 3 的用户界面 202。

[0103] 接口电路 920 还包括诸如调制解调器或网络接口卡之类的通信设备以促进经由网络 926 (例如,以太网连接、数字订户线路(DSL)、电话线、同轴电缆、蜂窝电话系统等等)与外部计算机的数据交换。

[0104] 处理器平台 900 还包括用于存储软件和数据的一个或多个大容量存储设备 928。此类大容量存储设备 928 的示例包括软盘驱动、硬驱动盘、光盘驱动、和数字多功能盘(DVD)驱动。大容量存储设备 928 可以实现应用测试器 204 (例如,以存储测试下软件应用 208)、缺陷重新产生器 206 (例如,以存储脚本 212)、缺陷验证器 312 (例如,以存储脚本 212)、脚本记录器 316 (例如,以存储日志 320)、和 / 或脚本生成器 318 (例如,以存储(一个或多个)所生成的脚本和 / 或存储脚本 212) 中的一个或多个。

[0105] 图 6-8 的经编码指令 932 可以被存储在大容量存储设备 928 中、易失性存储器 914 中、非易失性存储器 916 中、和 / 或诸如 CD 或 DVD 之类的可移除存储介质上。

[0106] 上文所述的示例性方法、装置、和制品提供了对软件缺陷的快速且高效的验证。对比于已知的软件缺陷验证的手动方法,本文所公开的示例性方法、装置、和制品更可靠,原因在于他们自动地重新产生导致软件缺陷的报告步骤而避免了会发生在手动过程期间的重新产生中的差错的可能性。因而,方法、装置、和制品通过使得更多的资源分配到开发和 / 或测试而不是使用先前的方法那样分配来准许更高质量的软件应用的开发。

[0107] 尽管在本文中已经描述了某些示例性方法、装置、和制品,但本专利的覆盖范围并不限于此。相反,本专利覆盖了公平落入本专利的权利要求范围内的所有方法、装置和制品。

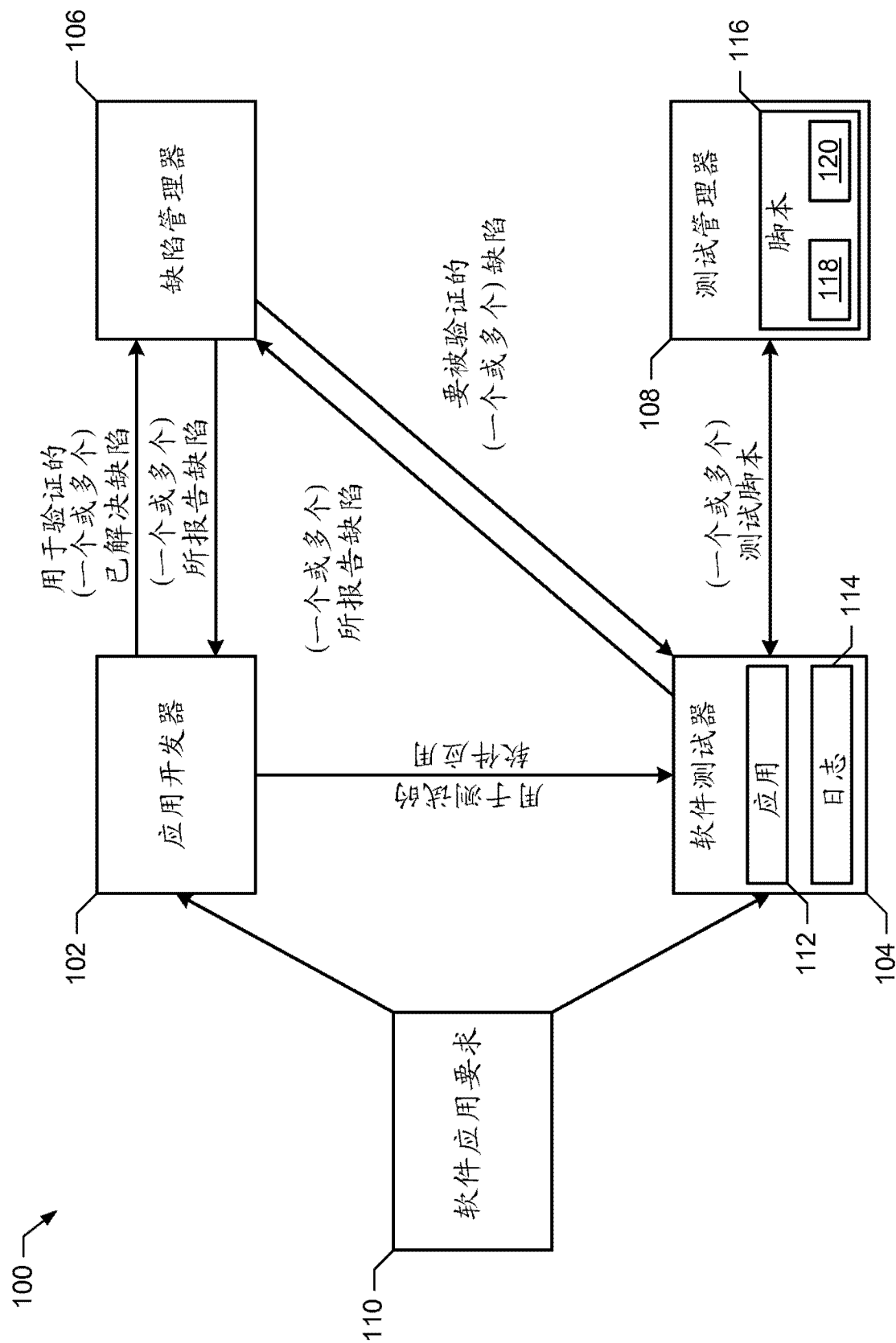


图 1

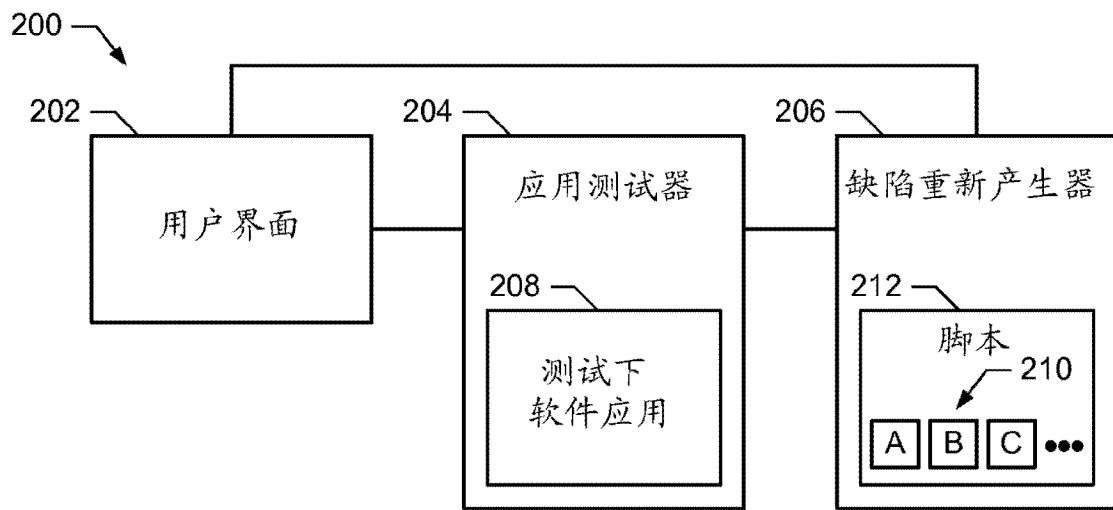


图 2

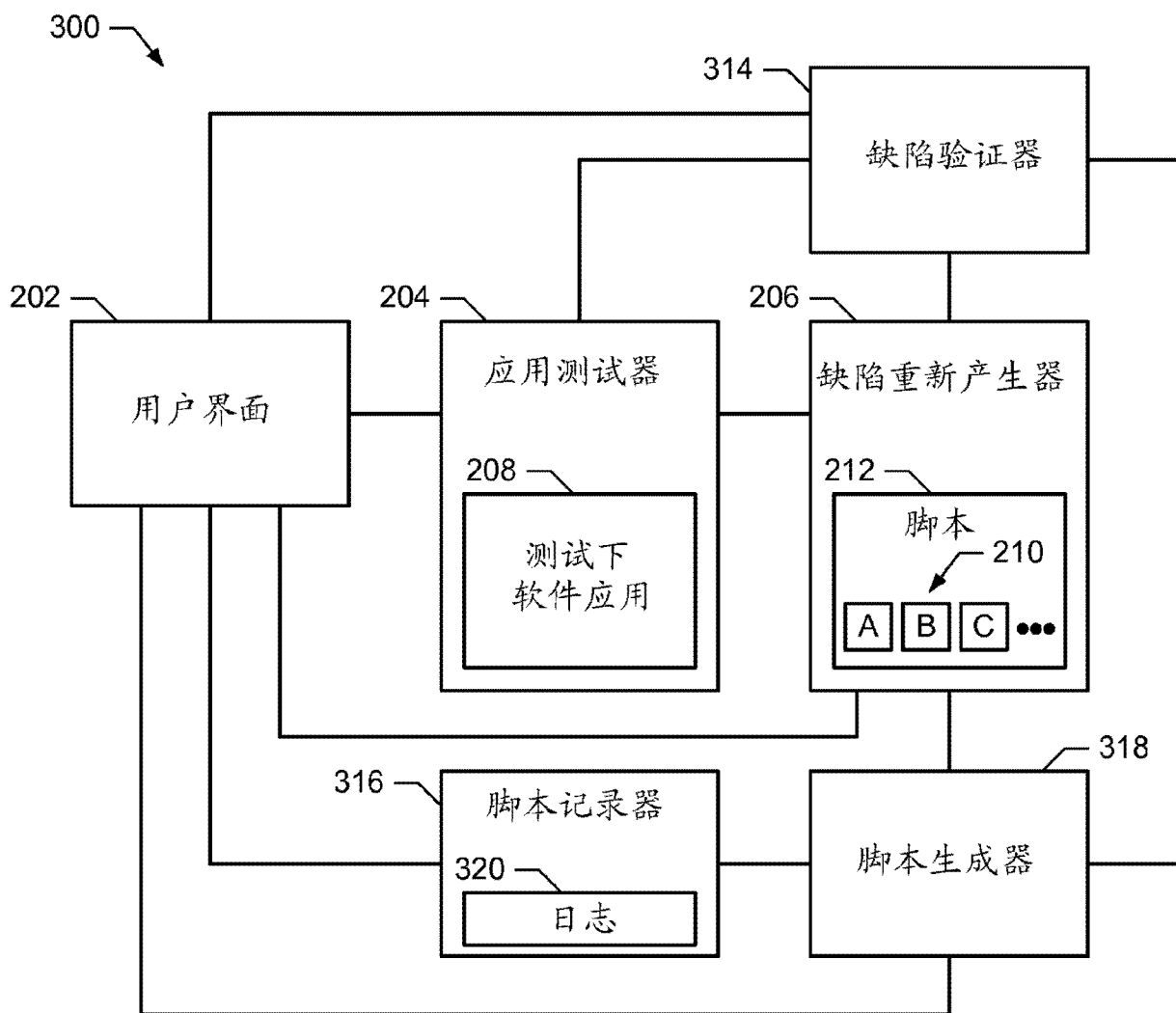


图 3

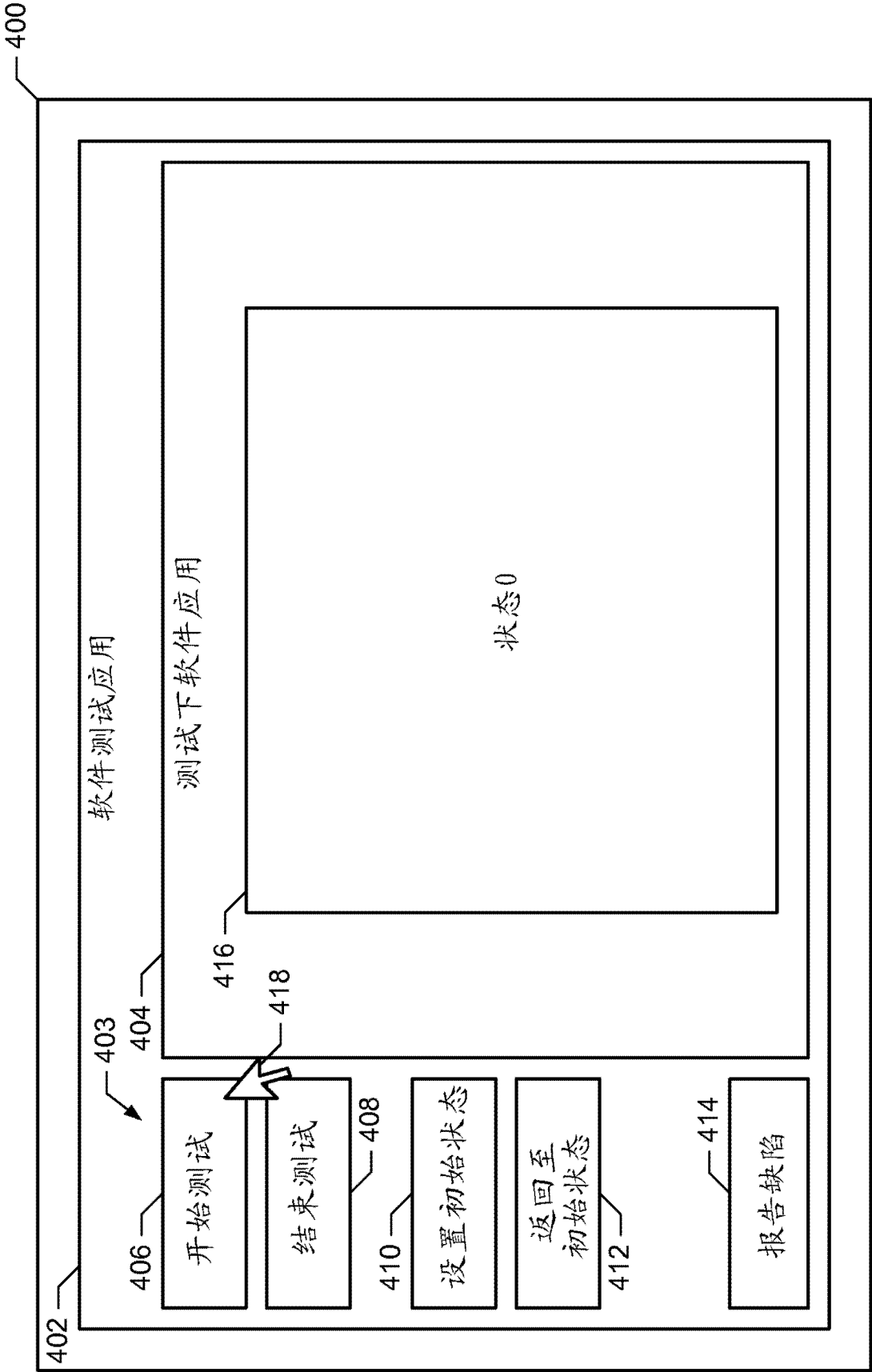


图 4A

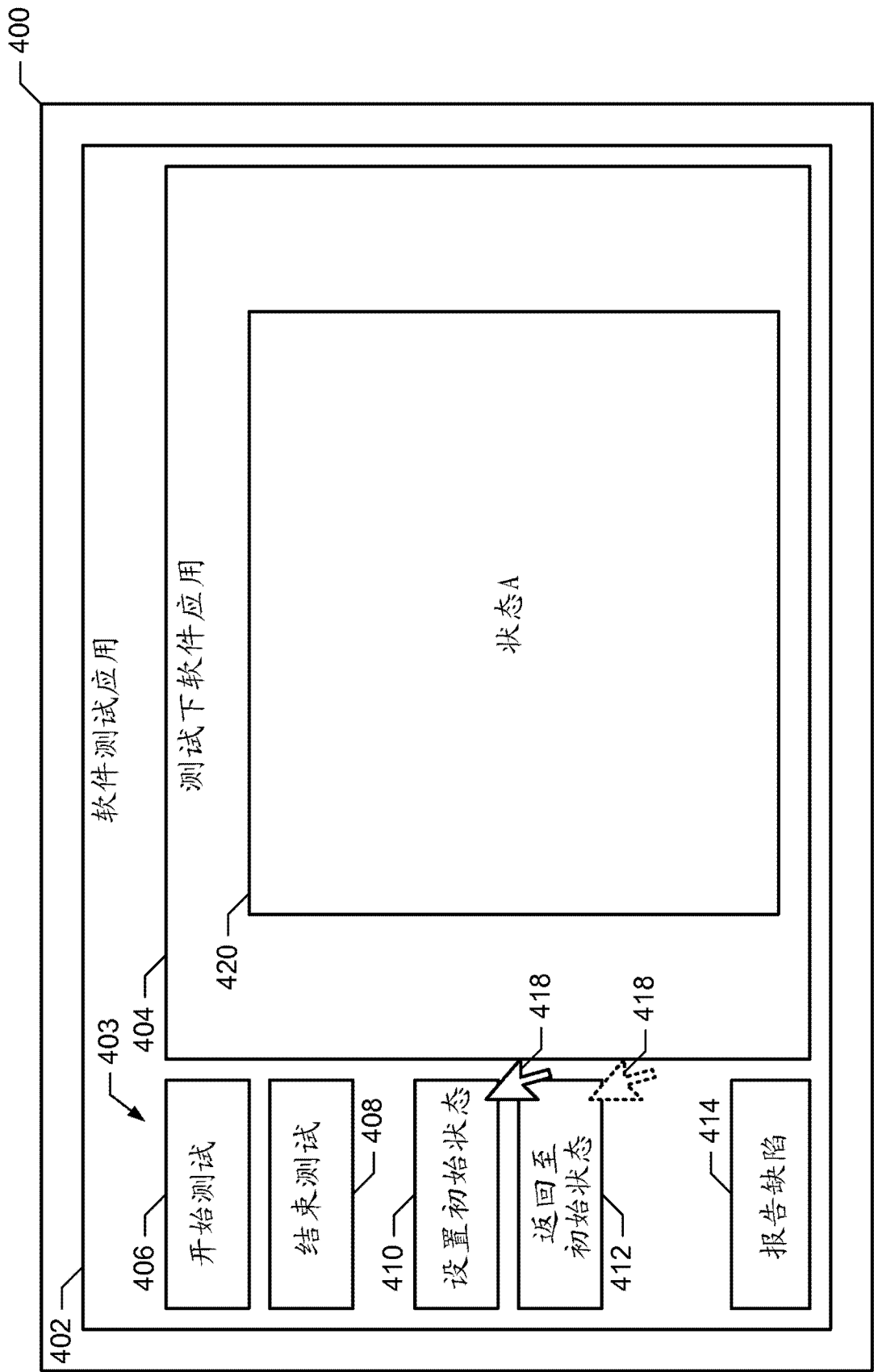


图 4B

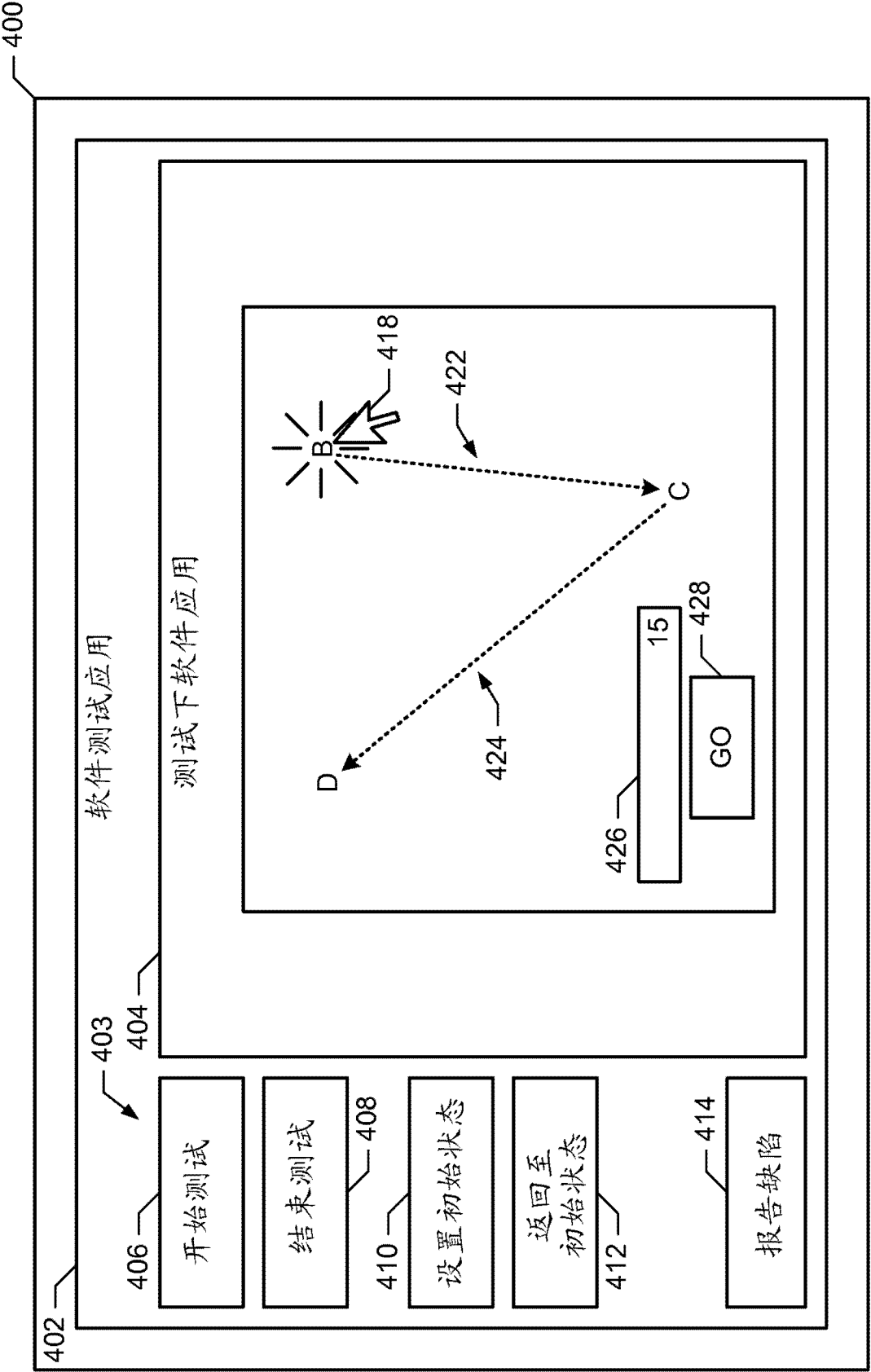


图 4C

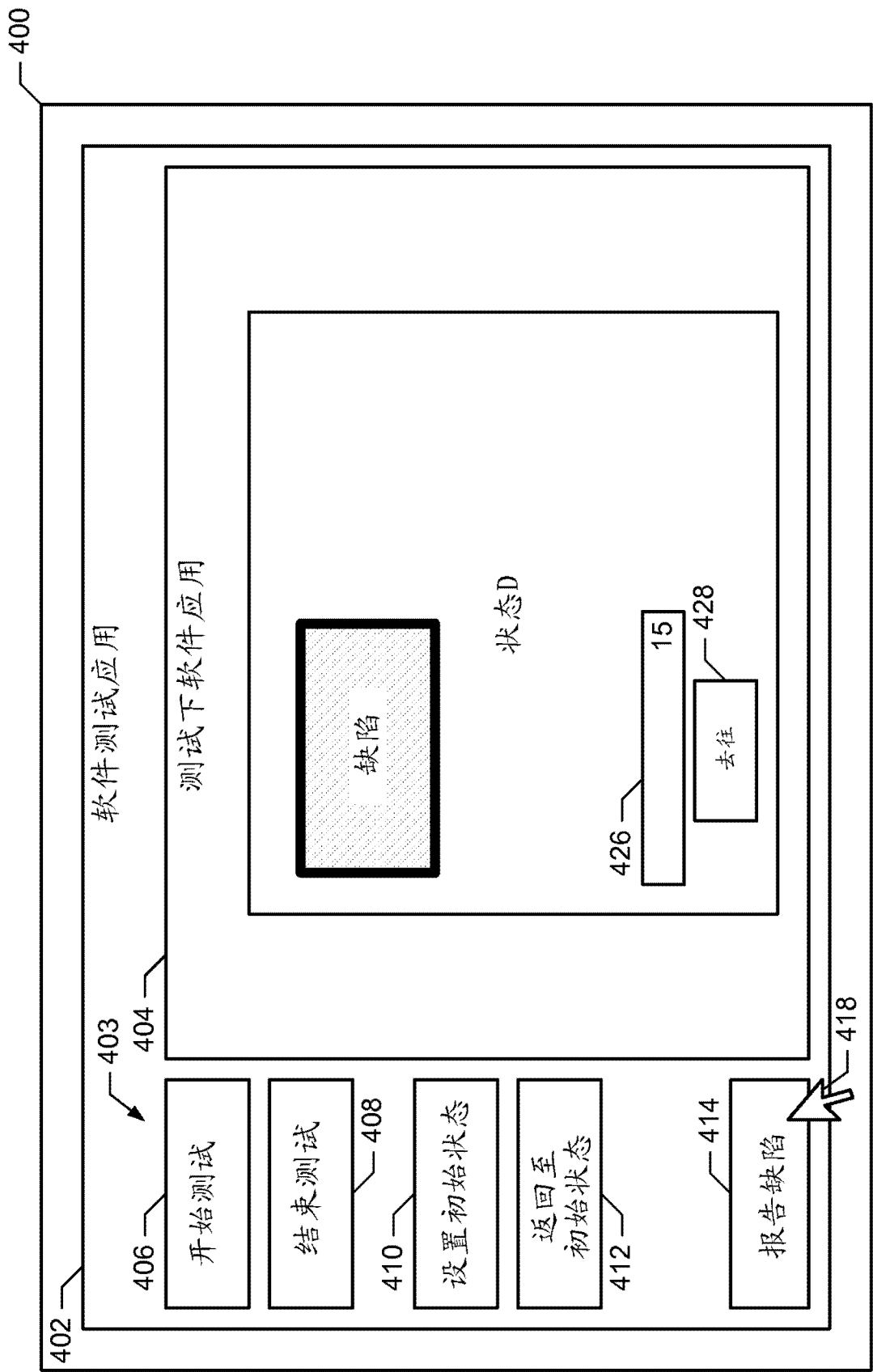


图 4D

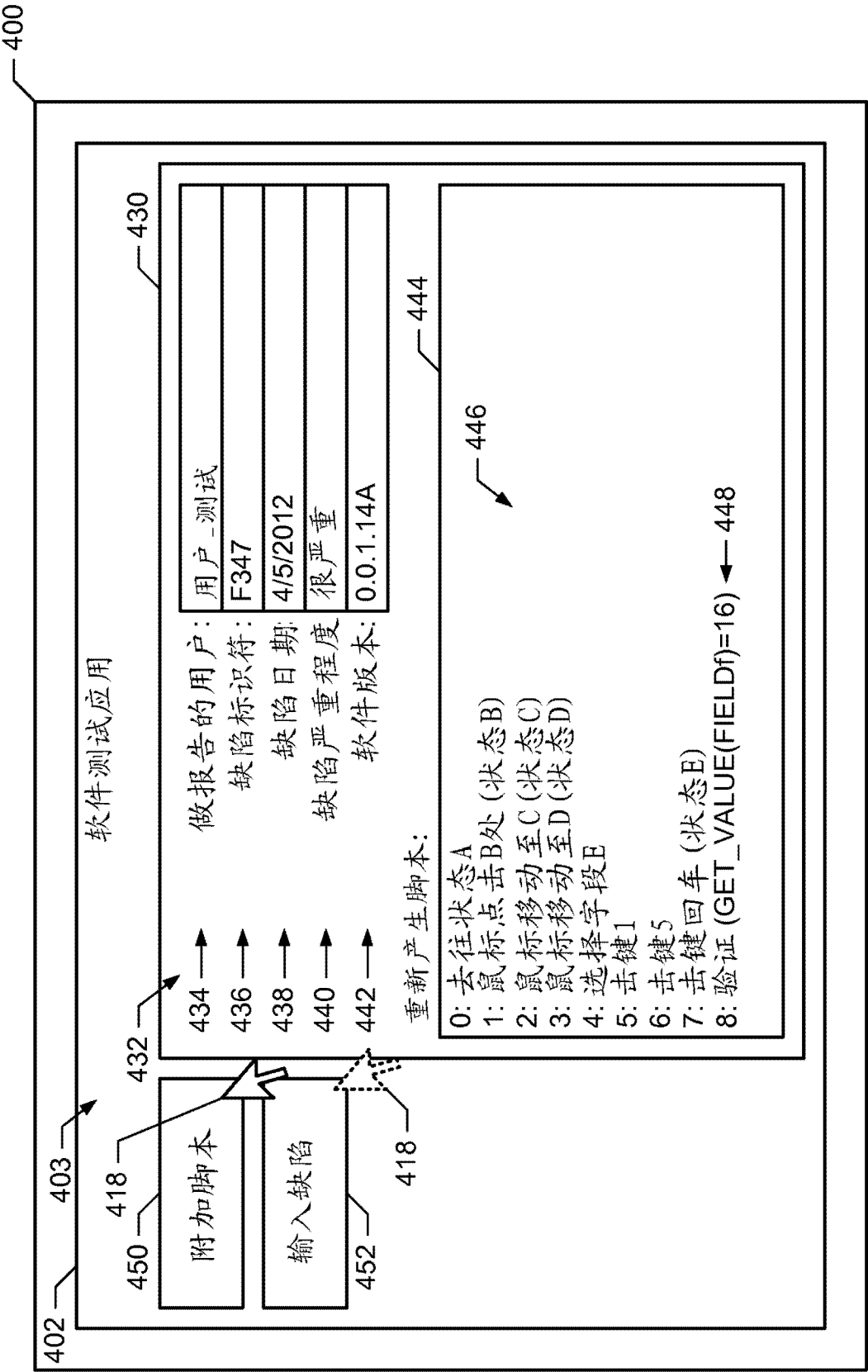


图 4E

24

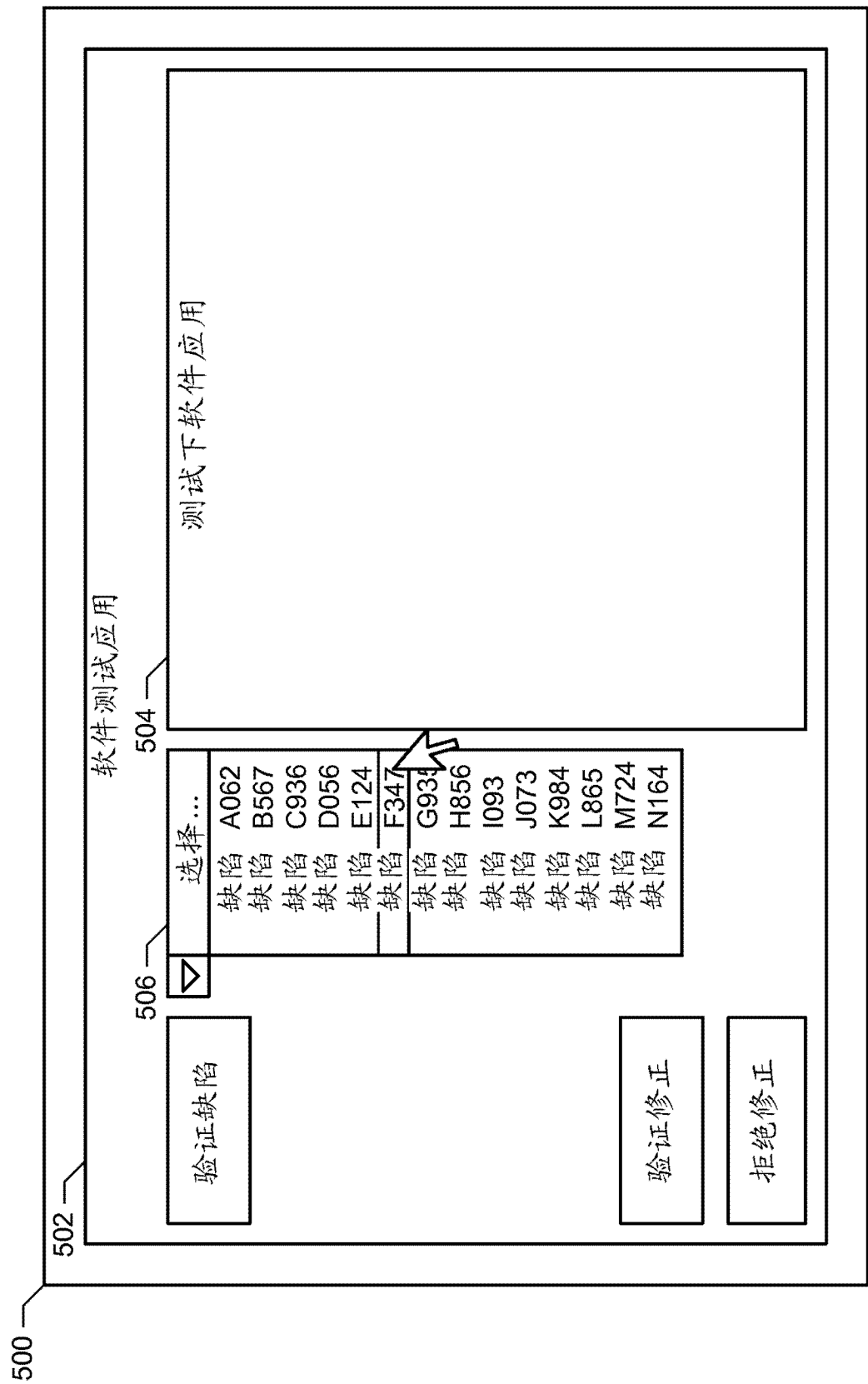


图 5A

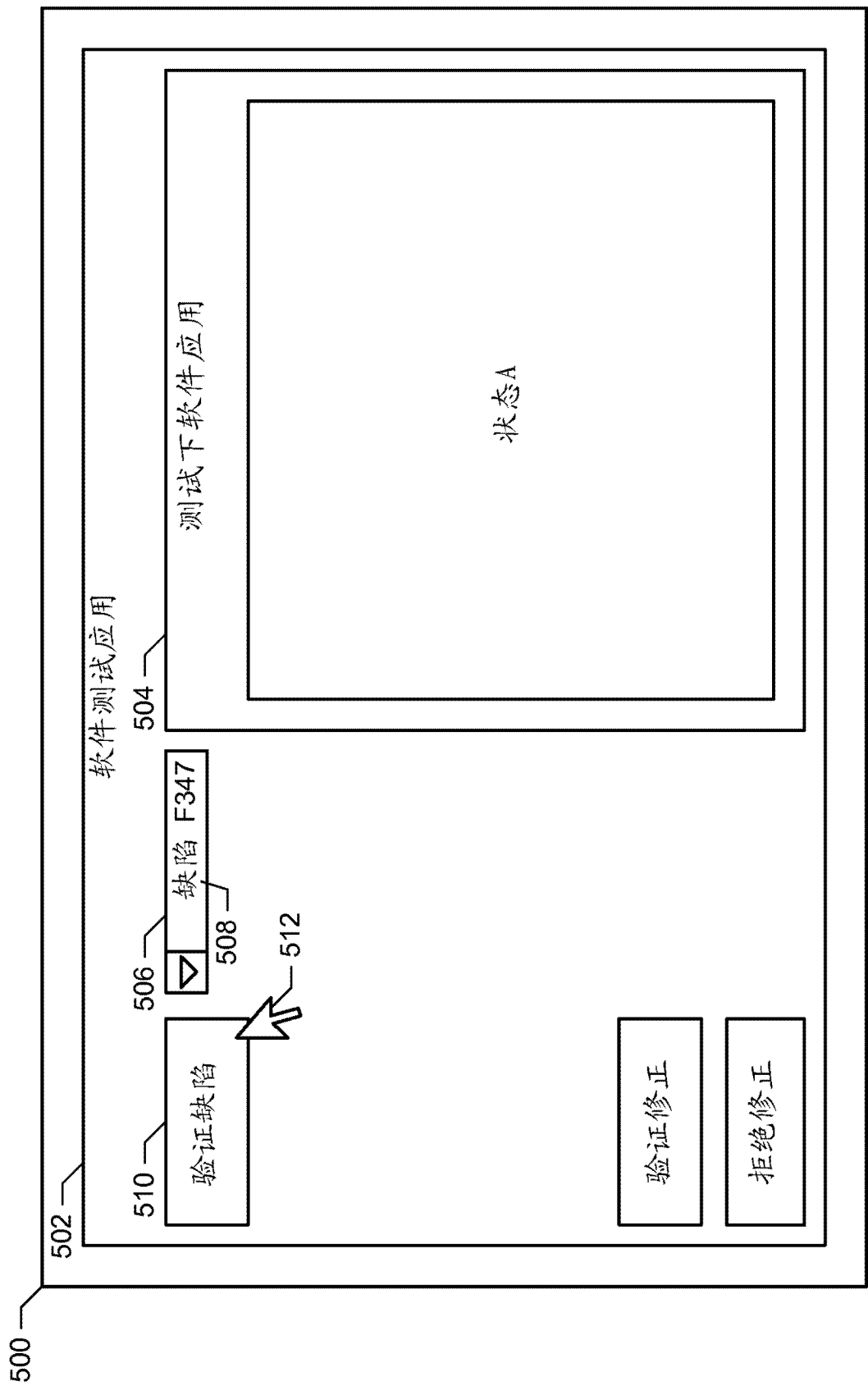


图 5B

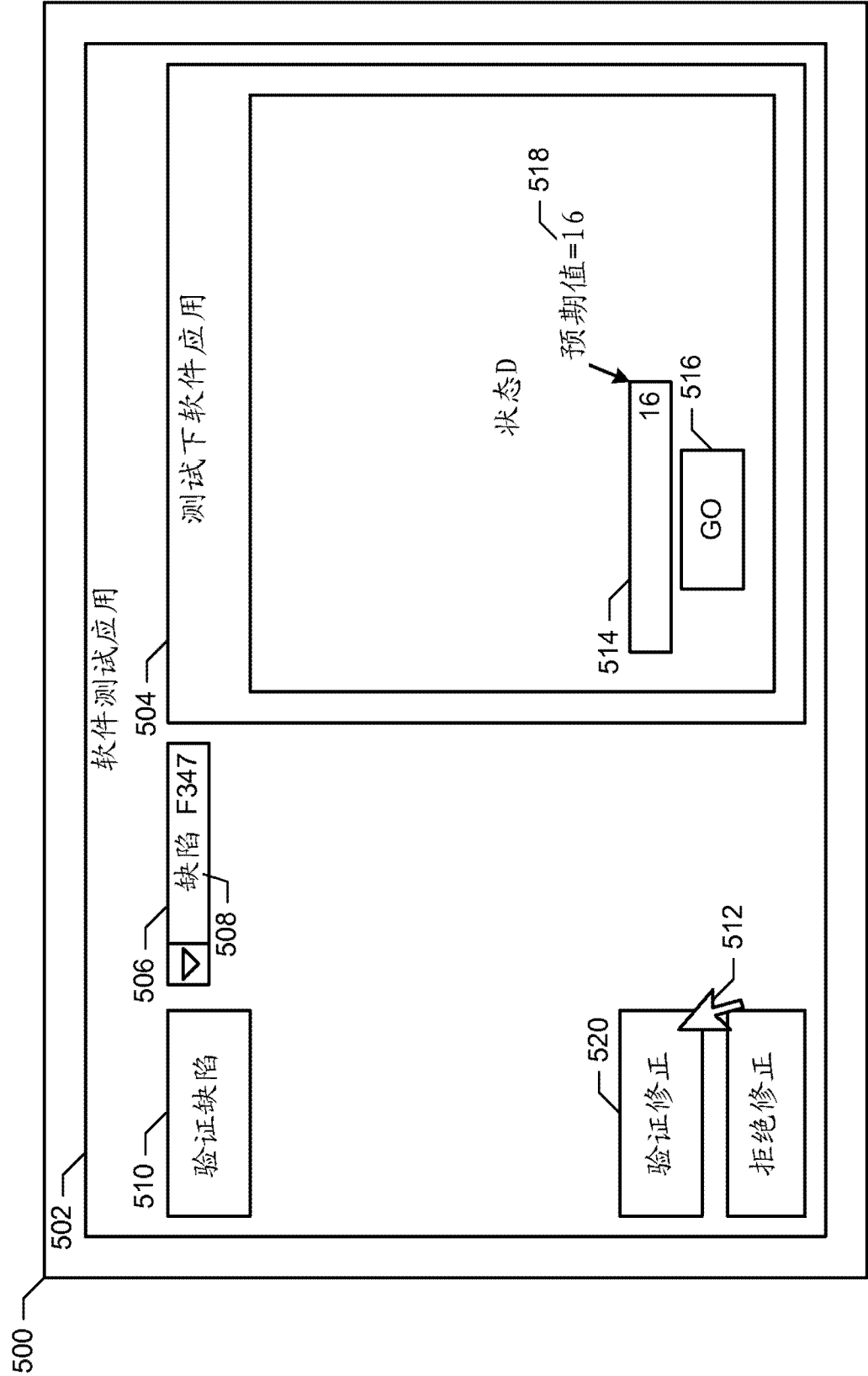


图 5C

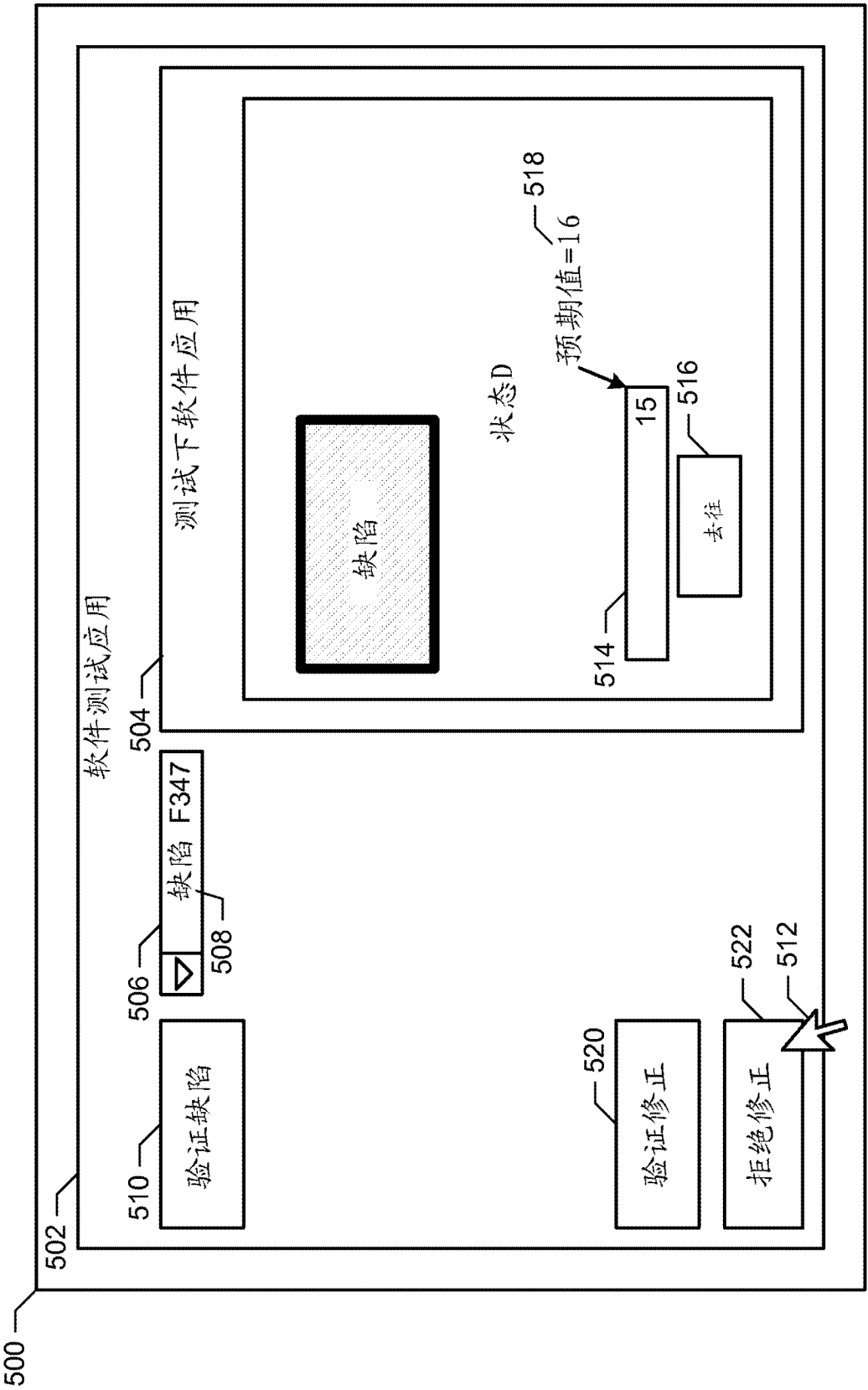


图 5D

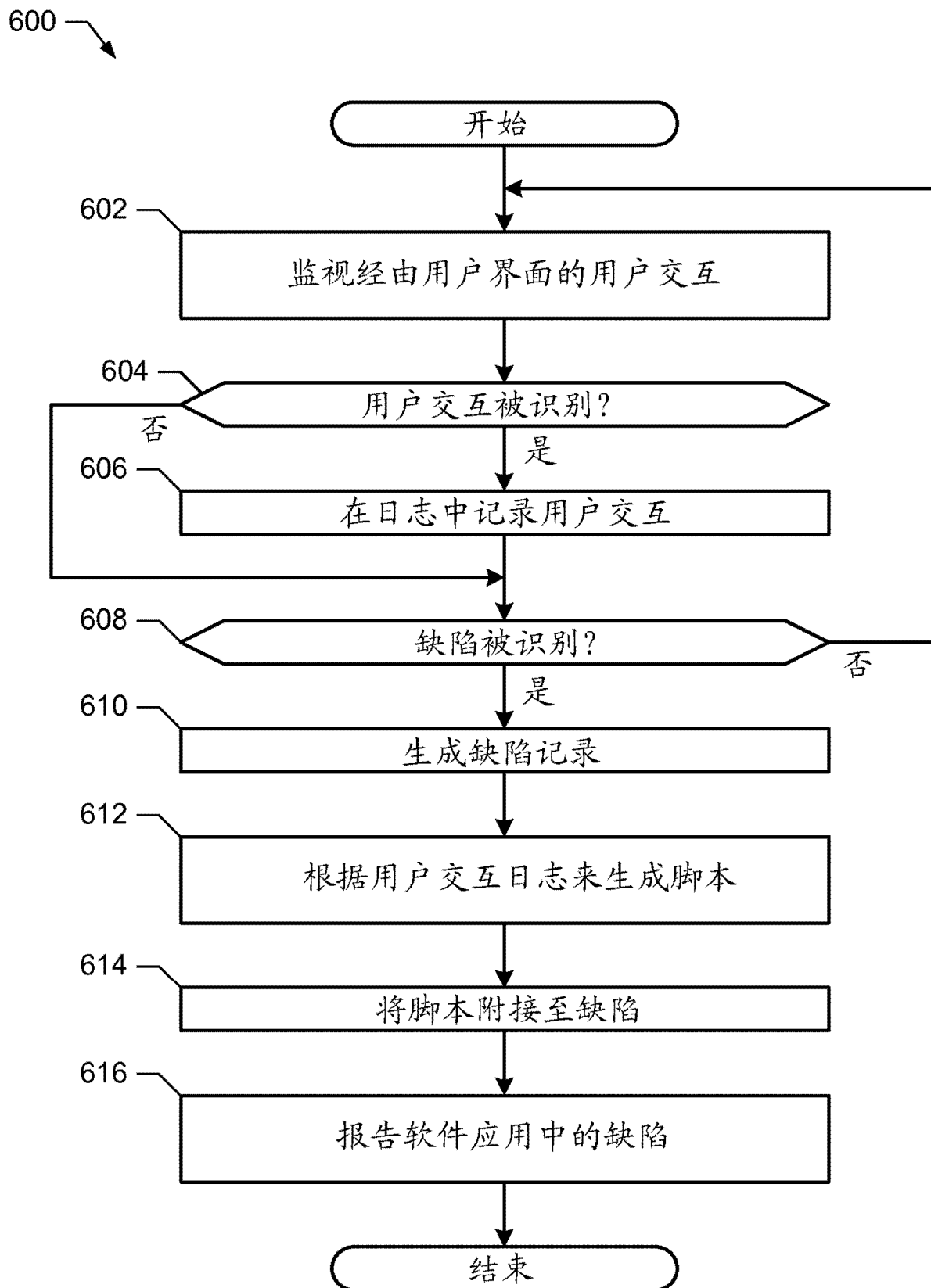


图 6

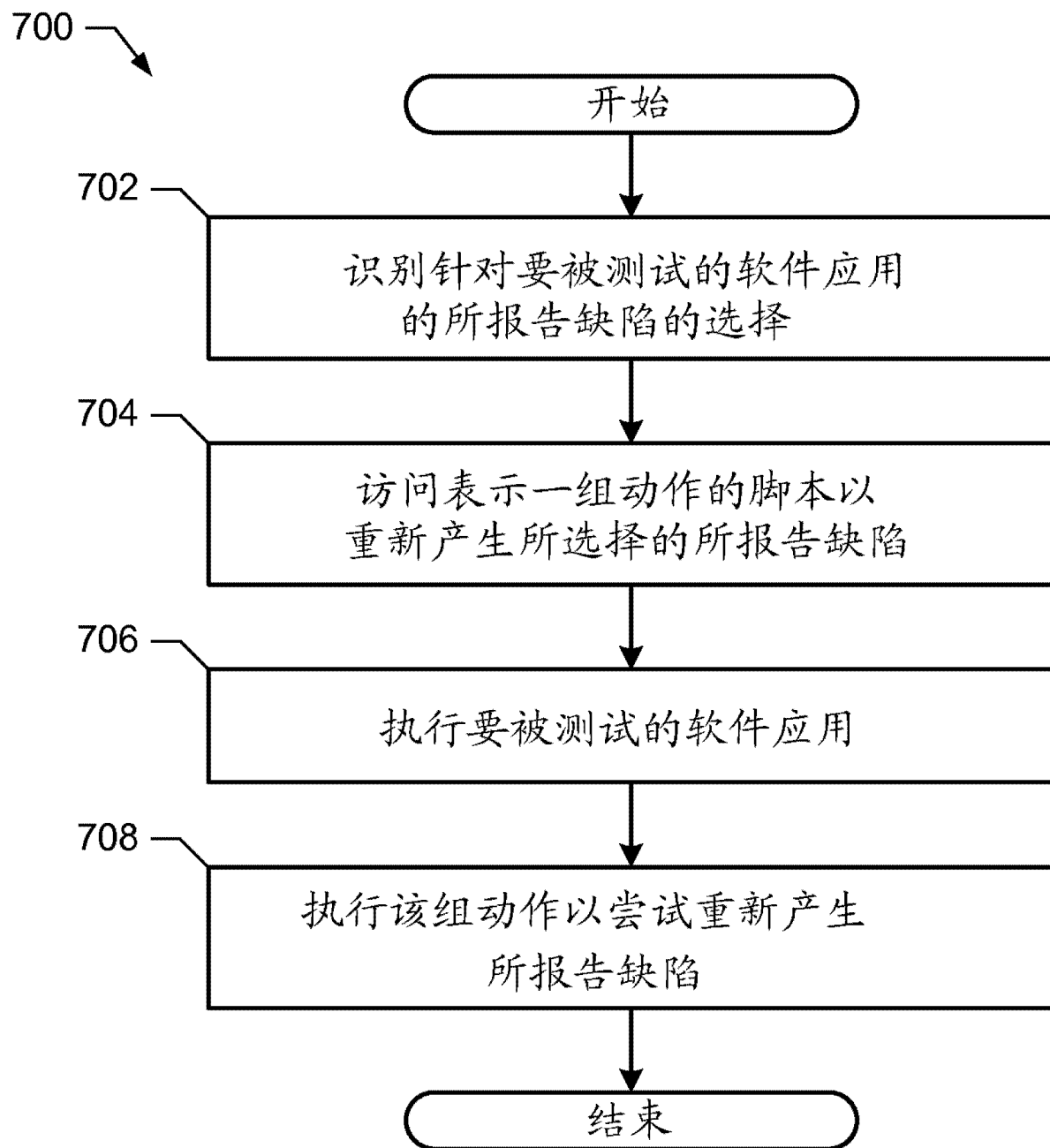


图 7

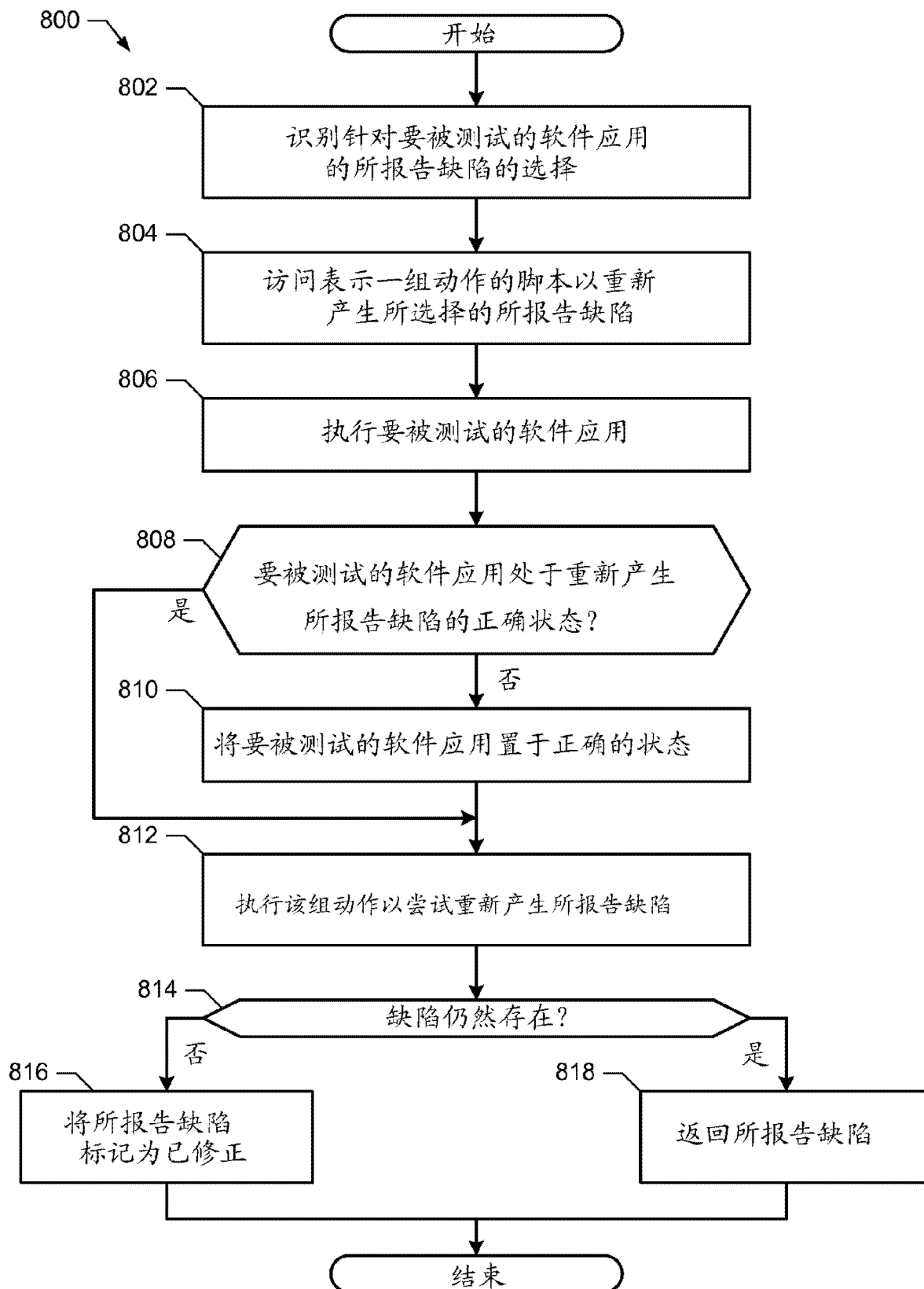


图 8

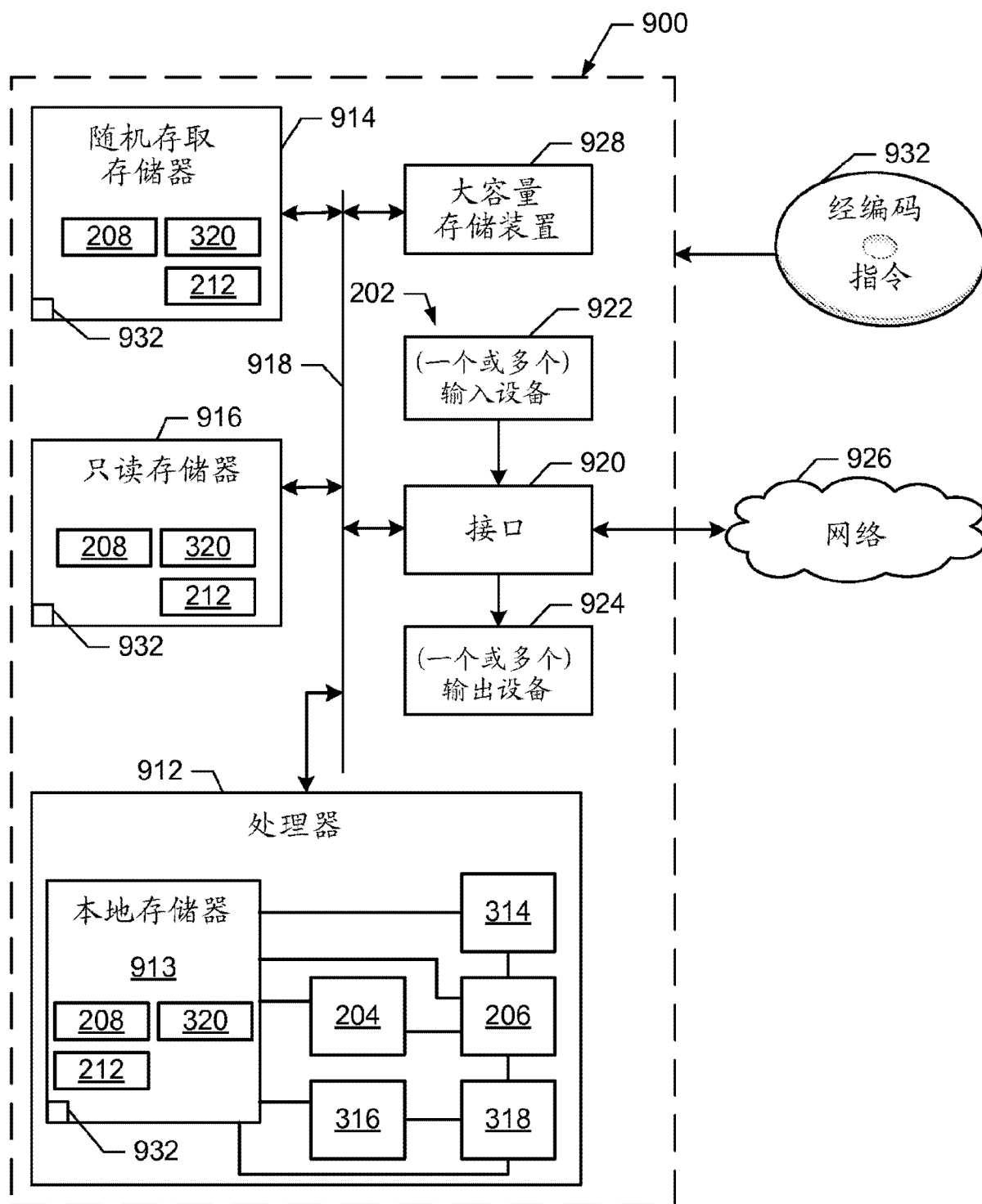


图 9