(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 118606160 A
(43)申请公布日 2024.09.06

(21)申请号 202310245593.7

(22)申请日 2023.03.06

(71)申请人 深圳市腾讯网络信息技术有限公司
地址 518102 广东省深圳市宝安区新安街道新安六路御景湾花园1栋101、201A、301、401之二楼201A-K49

(72)发明人 李凤兰 张戴克 陈强 王涵

(74)专利代理机构 北京三高永信知识产权代理有限责任公司 11138
专利代理师 张所明

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

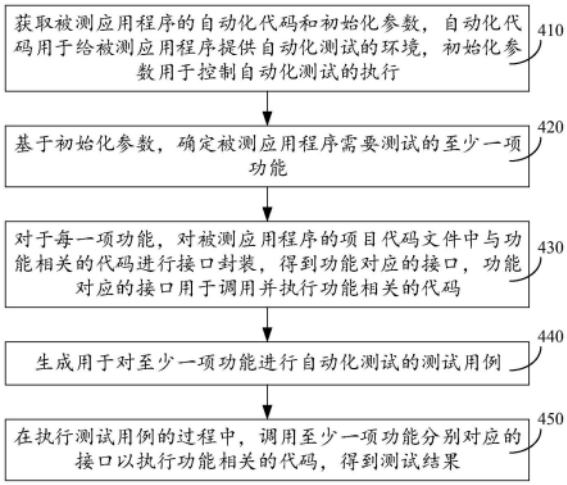
权利要求书3页 说明书16页 附图13页

(54)发明名称

自动化测试方法、装置、设备及存储介质

(57)摘要

本申请公开了一种自动化测试方法、装置、设备及存储介质，涉及计算机技术领域。所述方法包括：获取被测应用程序的自动化代码和初始化参数，自动化代码用于给被测应用程序提供自动化测试的环境；基于初始化参数，确定被测应用程序需要测试的至少一项功能；对于每一项功能，对被测应用程序的项目代码文件中与功能相关的代码进行接口封装，得到功能对应的接口；生成用于对至少一项功能进行自动化测试的测试用例；在执行测试用例的过程中，调用至少一项功能分别对应的接口以执行功能相关的代码，得到测试结果。本申请整个自动化测试流程均可以在同一个设备中自行完成，无需其他设备的干预，因此，提升了自动化测试的效率。



CN 118606160 A

1.一种自动化测试方法,其特征在于,所述方法包括:

获取被测应用程序的自动化代码和初始化参数,所述自动化代码用于给所述被测应用程序提供自动化测试的环境,所述初始化参数用于控制所述自动化测试的执行;

基于所述初始化参数,确定所述被测应用程序需要测试的至少一项功能;

对于每一项功能,对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装,得到所述功能对应的接口,所述功能对应的接口用于调用并执行所述功能相关的代码;

生成用于对所述至少一项功能进行自动化测试的测试用例;

在执行所述测试用例的过程中,调用所述至少一项功能分别对应的接口以执行所述功能相关的代码,得到测试结果。

2.根据权利要求1所述的方法,其特征在于,所述初始化参数中包括第一类型参数,所述第一类型参数用于确定所述需要测试的至少一项功能;

所述基于所述初始化参数,确定所述被测应用程序需要测试的至少一项功能,包括:

根据所述第一类型参数中的参数内容,确定所述被测应用程序需要测试的至少一项功能。

3.根据权利要求1所述的方法,其特征在于,所述初始化参数中包括第二类型参数,所述第二类型参数用于指示测试用例的生成方式;

所述生成用于对所述至少一项功能进行自动化测试的测试用例,包括:

在所述第二类型参数为第一数值的情况下,根据指定用例文件,生成所述测试用例,所述指定用例文件是执行过自动化测试并保存下来的用例文件;

或者,

在所述第二类型参数为第二数值的情况下,根据确定的所述需要测试的至少一项功能,生成所述测试用例;

其中,所述第一数值与所述第二数值不同。

4.根据权利要求3所述的方法,其特征在于,所述根据指定用例文件,生成所述测试用例,包括:

对所述指定用例文件进行解密及解析,得到解密及解析后的文件;

对所述解密及解析后的文件进行用例初始化,得到所述测试用例。

5.根据权利要求3所述的方法,其特征在于,所述根据确定的所述需要测试的至少一项功能,生成所述测试用例,包括:

确定所述需要测试的至少一项功能分别对应的接口;

根据所述自动化代码中携带的用例格式信息,基于所述需要测试的至少一项功能分别对应的接口,生成所述测试用例。

6.根据权利要求1所述的方法,其特征在于,所述项目代码文件包括第一代码文件和第二代码文件,其中,所述第一代码文件是使用第一编程语言编写的代码文件,所述第二代码文件是使用第二编程语言编写的代码文件,所述第一编程语言和所述第二编程语言不同,所述第一代码文件和所述第二代码文件分别实现所述被测应用程序的不同功能;

所述对于每一项功能,对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装,得到所述功能对应的接口,包括:

将所述第一代码文件中与第一类型功能相关的代码进行接口封装，得到所述功能对应的接口；

将所述第二代码文件中与第二类型功能相关的代码进行接口封装，得到所述功能对应的接口，所述第一类型功能和所述第二类型功能是所述被测应用程序所提供的不同的功能。

7.根据权利要求1所述的方法，其特征在于，所述测试用例的数量为至少一个，所述方法还包括：

在所述测试用例执行之前，基于所述自动化代码，确定用例执行控制逻辑，所述用例执行控制逻辑包括以下至少之一：用例依赖情况信息、执行函数信息、跳转信息、主流程验证信息、重试指示信息、保存方式信息；

基于所述用例执行控制逻辑，执行所述测试用例。

8.根据权利要求7所述的方法，其特征在于，所述基于所述用例执行控制逻辑，执行所述测试用例，包括：

在执行所述测试用例之前，根据所述用例执行控制逻辑包括的所述用例依赖情况信息所指示的所述测试用例对应的依赖用例的执行情况，确定是否执行所述测试用例；

在执行所述测试用例的过程中，根据所述用例执行控制逻辑包括的所述跳转信息，在所述测试用例中的第一步骤执行成功的情况下，从所述第一步骤跳转至第二步骤开始执行，所述第一步骤和所述第二步骤之间的跳转关系存储在所述跳转信息中，所述第二步骤是所述测试用例中除去所述第一步骤以外的其他步骤，或其他测试用例中的步骤；

在所述测试用例执行失败的情况下，根据所述用例执行控制逻辑包括的所述重试指示信息，对所述测试用例进行重新测试。

9.根据权利要求1所述的方法，其特征在于，所述测试结果中包括第一类型数据和第二类型数据，所述第一类型数据反映所述测试用例是否通过，所述第二类型数据表征所述测试用例在执行过程中的性能数据；

所述第一类型数据用于生成冒烟报告，所述第二类型数据用于生成性能报告。

10.一种自动化测试装置，其特征在于，所述装置包括：

获取模块，用于获取被测应用程序的自动化代码和初始化参数，所述自动化代码用于给所述被测应用程序提供自动化测试的环境，所述初始化参数用于控制所述自动化测试的执行；

功能确定模块，用于基于所述初始化参数，确定所述被测应用程序需要测试的至少一项功能；

接口封装模块，用于对于每一项功能，对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装，得到所述功能对应的接口，所述功能对应的接口用于调用并执行所述功能相关的代码；

用例生成模块，用于生成用于对所述至少一项功能进行自动化测试的测试用例；

用例执行模块，用于在执行所述测试用例的过程中，调用所述至少一项功能分别对应的接口以执行所述功能相关的代码，得到测试结果。

11.一种计算机设备，其特征在于，所述计算机设备包括处理器和存储器，所述存储器中存储有计算机程序，所述计算机程序由所述处理器加载并执行以实现如上述权利要求1

至9任一项所述的方法。

12.一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程序，所述计算机程序由处理器加载并执行以实现如上述权利要求1至9任一项所述的方法。

13.一种计算机程序产品，其特征在于，所述计算机程序产品包括计算机程序，所述计算机程序存储在计算机可读存储介质中，处理器从所述计算机可读存储介质读取并执行所述计算机程序，以实现如上述权利要求1至9任一项所述的方法。

自动化测试方法、装置、设备及存储介质

技术领域

[0001] 本申请实施例涉及计算机技术领域，特别涉及一种自动化测试方法、装置、设备及存储介质。

背景技术

[0002] 为了避免应用程序上线后出现问题难易解决，开发测试人员会在应用程序上线之前，对该应用程序的项目文件代码进行测试以提前发现代码中的问题。

[0003] 相关技术中，针对应用程序的代码测试主要是UI(User Interface，用户界面)自动化测试，UI自动化测试中，通常需要开发测试人员使用Python等脚本语言编写测试用例，然后向对应的运行有被测应用程序的终端发送测试用例，使得终端可以执行该测试用例，从而实现自动化测试。

[0004] 然而，相关技术中的UI自动化测试，需要开发测试人员通过PC(Personal Computer，个人计算机)等开发测试设备来外围驱动终端执行自动化测试，因此，自动化测试效率较低。

发明内容

[0005] 本申请实施例提供了一种自动化测试方法、装置、设备及存储介质。所述技术方案如下：

[0006] 根据本申请实施例的一个方面，提供了一种用于自动化测试方法，所述方法包括：

[0007] 获取被测应用程序的自动化代码和初始化参数，所述自动化代码用于给所述被测应用程序提供自动化测试的环境，所述初始化参数用于控制所述自动化测试的执行；

[0008] 基于所述初始化参数，确定所述被测应用程序需要测试的至少一项功能；

[0009] 对于每一项功能，对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装，得到所述功能对应的接口，所述功能对应的接口用于调用并执行所述功能相关的代码；

[0010] 生成用于对所述至少一项功能进行自动化测试的测试用例；

[0011] 在执行所述测试用例的过程中，调用所述至少一项功能分别对应的接口以执行所述功能相关的代码，得到测试结果。

[0012] 根据本申请实施例的一个方面，提供了一种自动化测试装置，所述装置包括：

[0013] 获取模块，用于获取被测应用程序的自动化代码和初始化参数，所述自动化代码用于给所述被测应用程序提供自动化测试的环境，所述初始化参数用于控制所述自动化测试的执行；

[0014] 功能确定模块，用于基于所述初始化参数，确定所述被测应用程序需要测试的至少一项功能；

[0015] 接口封装模块，用于对于每一项功能，对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装，得到所述功能对应的接口，所述功能对应的接口用于

调用并执行所述功能相关的代码；

[0016] 用例生成模块，用于生成用于对所述至少一项功能进行自动化测试的测试用例；

[0017] 用例执行模块，用于在执行所述测试用例的过程中，调用所述至少一项功能分别对应的接口以执行所述功能相关的代码，得到测试结果。

[0018] 根据本申请实施例的一个方面，提供了一种计算机设备，所述计算机设备包括处理器和存储器，所述存储器中存储有计算机程序，所述计算机程序由所述处理器加载并执行以实现上述方法。

[0019] 根据本申请实施例的一个方面，提供了一种计算机可读存储介质，所述可读存储介质中存储有计算机程序，所述计算机程序由处理器加载并执行以实现上述方法。

[0020] 根据本申请实施例的一个方面，提供了一种计算机程序产品，该计算机程序产品包括计算机程序，该计算机程序存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质读取该计算机程序，处理器执行该计算机程序，使得该计算机设备执行上述方法。

[0021] 本申请实施例提供的技术方案可以包括如下有益效果：

[0022] 通过针对被测应用程序的自动化代码和初始化参数，可以拉起被测应用程序的客户端，并在客户端中生成自动化测试环境并控制开始执行自动化测试。一方面来说，通过初始化参数来明确被测应用程序需要测试的功能，并根据需要测试的功能来自动化生成测试用例。另一方面，对被测应用程序的项目代码文件进行接口封装，使得测试用例在执行过程中，可以直接调用接口，来快速地完成测试用例的执行。本申请实施例提供的技术方案，在接收到自动化代码和初始化参数之后，整个自动化测试流程均可以在同一个计算机设备中自行完成，无需其他开发测试设备的干预，因此，提升了自动化测试的效率。另外，由于针对需要测试的功能进行接口封装，在执行测试用例时，只需要快速调用对应的接口执行即可，缩短了测试所需要的时间。

附图说明

[0023] 图1是本申请一个实施例提供的方案实施环境的示意图；

[0024] 图2是本申请一个实施例提供的自动化测试方法的示意图；

[0025] 图3是本申请一个实施例提供的自动化测试方法的框图；

[0026] 图4是本申请一个实施例提供的自动化测试方法的流程图；

[0027] 图5是本申请另一个实施例提供的自动化测试方法的流程图；

[0028] 图6是本申请一个实施例提供的用例生成及用例执行的示意图；

[0029] 图7是本申请另一个实施例提供的自动化测试方法的流程图；

[0030] 图8是本申请一个实施例提供的用例执行控制逻辑的示意图；

[0031] 图9是本申请一个实施例提供的冒烟报告的示意图；

[0032] 图10是本申请一个实施例提供的自动化测试框架的示意图；

[0033] 图11是本申请一个实施例提供的Lua自动化代码以及自动化接口的代码示意图；

[0034] 图12是本申请一个实施例提供的生成用例以及用例文件的代码示意图；

[0035] 图13是本申请一个实施例提供的自动化测试装置的框图；

[0036] 图14是本申请另一个实施例提供的自动化测试装置的框图；

[0037] 图15是本申请一个实施例提供的计算机设备的结构框图。

具体实施方式

[0038] 为使本申请的目的、技术方案和优点更加清楚，下面将结合附图对本申请实施方式作进一步地详细描述。

[0039] 在介绍本申请技术方案之前，先对本申请涉及的一些名词进行解释说明。以下相关解释作为可选方案与本申请实施例的技术方案可以进行任意结合，其均属于本申请实施例的保护范围。本申请实施例包括以下内容中的至少部分内容。

[0040] 游戏引擎：游戏引擎就是集成了复杂功能的游戏开发软件，他们实现了复杂的底层逻辑，比如：物理系统，粒子系统，寻路系统，图形渲染等等。因此对于开发者来说不再需要具备太多专业而复杂的计算机专业知识，只需要进行简单的系统学习，便可以使用它们来进行游戏开发。Unity和UE4(Unreal Engine 4，虚幻引擎4)都是游戏引擎。

[0041] UE4：是一套跨平台的游戏引擎，可用于开发Windows、MacOS、Linux平台的单机游戏，或是ios、android移动设备的游戏。当然，除去游戏之外，UE4还可以用于其他类型应用程序的开发，例如建筑设计类等等。

[0042] Lua：一个小巧的脚本语言。其设计目的是为了通过灵活嵌入应用程序中从而为应用程序提供灵活的扩展和定制功能。Lua由标准C语言编写而成，几乎在所有操作系统和平台上都可以编译，运行。Lua并没有提供强大的库，这是由它的定位决定的。所以Lua不适合作为开发独立应用程序的语言。Lua有一个同时进行的JIT项目，提供在特定平台上的即时编译功能。Lua脚本可以很容易的被C/C++代码调用，也可以反过来调用C/C++的函数，这使得Lua在应用程序中可以被广泛应用。不仅仅作为扩展脚本，也可以作为普通的配置文件，并且更容易理解和维护。Lua由标准C语言编写而成，代码简洁优美，几乎在所有操作系统和平台上都可以编译，运行。一个完整的Lua解释器不过200k，在所有脚本引擎中，Lua的速度是最快的。这一切都决定了Lua是作为嵌入式脚本的最佳选择。很多应用程序、游戏使用Lua作为自己的嵌入式脚本语言，以此来实现可配置性、可扩展性。

[0043] Unlua：适用于UE的一个高度优化的Lua脚本解决方案。作为UE4特性丰富且高度优化lua插件，Unlua被应用在大量大型项目中。Unlua可以零胶水访问引擎反射体系的所有UCLASS(告知虚幻引擎生成类的反射数据)、UPROPERTY(使UCLASS或USTRUCT的成员变量可用作UPROPERTY，UPROPERTY允许变量被复制、被序列化，并可从蓝图中进行访问)、UFUNCTION(使UCLASS或USTRUCT的类方法可用作UFUNCTION)、USTRUCT(告知虚幻引擎生成结构体的反射数据)、UENUM(枚举)等。完备的静态导出方案，用于导出引擎反射系统之外的类(成员函数、成员变量)、全局函数、枚举。

[0044] Appium：是一个自动化测试开源工具，支持ios和android平台上的移动原生应用、移动Web应用和混合应用。Appium是一个跨平台工具，它允许测试人员使用同样的接口、基于不同的平台写自动化测试代码，大大增加了测试套件间代码的复用性。移动原生应用：是指那些用ios或者android SDK写的应用；移动web应用：是指那些使用移动浏览器访问的应用，Appium支持ios的safari和android上的chrome；混合应用：是指原生代码封装在网页视图(原生代码和web内容交互)。

[0045] 跨平台：可以简单理解为不同的操作系统，比如家用电脑使用最多的windows操作

系统,苹果电脑的MacOS操作系统,包括安卓手机的android系统,苹果手机的ios系统等等,这些不同设备因为他们的操作系统不一样就称为不同的平台。以前开发者开发一款游戏,为了能在不同的平台上使用,就必须得针对不同的平台进行多次开发。而跨平台的意思就是,只需要进行一次开发,通过Unity和UE提供的跨平台功能,可以让产品在各种不同平台上使用,并且不需要进行二次开发。

[0046] Python:是一种计算机编程语言,是一个高层次的结合了解释性、编译性、互动性和面向对象的脚本语言。

[0047] Gautomator:一种手游客户端自动化测试框架,在项目代码中集成SDK(Software Development Kit,软件开发工具包),在游戏外部与SDK通信,控制游戏执行测试。

[0048] 请参考图1,其示出了本申请一个实施例提供的方案实施环境的示意图。该方案实施环境可以包括:终端设备10和服务器20。

[0049] 终端设备10包括但不限于手机、平板电脑、智能语音交互设备、游戏主机、可穿戴设备、多媒体播放设备、PC、车载终端、智能家电等电子设备。终端设备10中可以安装被测应用程序的客户端。

[0050] 在本申请实施例中,上述被测应用程序可以是任何能够实现自动化测试的应用程序。可选地,被测应用程序可以是游戏类应用程序、互动娱乐类应用程序、社交类应用程序、仿真类应用程序、虚拟现实(Virtual Reality,简称VR)类应用程序、增强现实(Augmented Reality,简称AR)类应用程序等。可选地,游戏类应用程序还包括第一人称射击游戏(First-Person Shooting Game,简称FPS)、多人枪战类生存游戏、第三人称射击游戏(Third-Person Shooting Game,简称TPS)、多人在线战术竞技(Multiplayer Online Battle Arena,简称MOBA)游戏、策略游戏(Simulation Game,简称SLG)等等,本申请实施例对此不作限定。当然,对于不同的被测应用程序来说,其对应的被测应用程序的项目文件代码不同,相应地,需要进行自动化测试的被测应用程序的代码也不相同。本申请实施例对此不作限定。可选地,终端设备10中运行有上述被测应用程序的客户端。

[0051] 服务器20用于为终端设备10中的被测应用程序的客户端提供后台服务。例如,服务器20可以是独立的物理服务器,也可以是多个物理服务器构成的服务器集群或者分布式系统,还可以是提供云服务、云数据库、云计算、云函数、云存储、网络服务、云通信、中间件服务、域名服务、安全服务、CDN(Content Delivery Network,内容分发网络)、以及大数据和人工智能平台等基础云计算服务的云服务器,但并不局限于此。

[0052] 终端设备10和服务器20之间可通过网络进行互相通信。该网络可以是有线网络,也可以是无线网络。

[0053] 本申请实施例提供的方法,各步骤的执行主体可以是计算机设备。计算机设备可以是任何具备数据的存储和处理能力的电子设备。例如,计算机设备可以是图1中的服务器20,可以是图1中的终端设备10,也可以是除终端设备10和服务器20以外的另一设备。

[0054] 在一些实施例中,计算机设备还可以是第一计算机设备或者第二计算机设备。在一些实施例中,第一计算机设备用于接收被测应用程序的项目代码文件,并执行上述自动化测试方法。可选地,第二计算机设备用于给第一计算机设备发送执行自动化测试所需要的自动化代码以及初始化参数。在一些实施例中,第一计算机设备是第一终端设备或者第一服务器中的至少之一。可选地,第二计算机设备是第二终端设备或者第二服务器中的至

少之一。第一终端设备和二终端设备是相同或者不同的终端设备，第一服务器和第二服务器是相同或者不同的服务器。

[0055] 请参考图2，其示出了本申请一个实施例提供的自动化测试方法的示意图。该方法各步骤的执行主体可以是第一计算机设备或者第二计算机设备，第一计算机设备和第二计算机设备可以是相同或者不同的设备，本申请对此不作限定。为了便于描述，本实施例中第一计算机设备认为是开发测试设备，第二计算机设备认为是终端设备。

[0056] 在一些实施例中，在开发测试设备上对被测应用程序的项目文件代码进行开发调试。可选地，被测应用程序是游戏应用程序，开发测试设备上运行有虚幻引擎，通过虚幻引擎对游戏应用程序的项目文件代码进行开发调试。可选地，在项目代码文件的开发调试过程中，需要使用对游戏应用程序的运行进行自动化测试。可选地，利用虚幻引擎生成该游戏应用程序的安装包。该安装包至少可以在android或者ios系统上安装成为被测应用程序的客户端。当然，开发测试设备需要和真机或者模拟器建立连接。其中，真机可以认为是真实的终端设备，而模拟器认为是虚拟的终端设备。可选地，在建立连接之后，开发测试设备向对应的终端设备(真机或者模拟器)发送被测应用程序的项目代码文件。当然，也可以认为开发测试设备向对应的终端设备发送被测应用程序的项目代码文件所对应的安装包。可选地，终端设备根据项目文件代码，生成被测应用程序的客户端。可选地，终端设备在接收到被测应用程序的安装包之后，下载安装被测应用程序的客户端。

[0057] 在另一些实施例中，开发测试设备需要生成被测应用程序的自动化代码，并确定初始化参数。其中，自动化代码可以认为是驱动终端设备上的客户端，做好自动化测试准备的代码。而初始化参数可以看作是控制终端设备是否执行自动化以及如何执行自动化的参数。

[0058] 在以上步骤完成之后，本申请实施例提供的技术方案还包括如下几个步骤：

[0059] 步骤S1，开发测试设备向终端设备发送自动化代码。

[0060] 步骤S2，开发测试设备向终端设备发送初始化参数。

[0061] 步骤S3，终端设备确定被测应用程序需要测试的功能。

[0062] 步骤S4，终端设备对项目文件代码进行接口封装。

[0063] 步骤S5，终端设备针对需要测试的功能生成测试用例。

[0064] 步骤S6，终端设备调用接口执行测试用例，得到测试结果。

[0065] 终端设备在得到测试结果之后，将测试结果发送给开发测试设备，开发测试设备根据测试结果，生成冒烟报告或者性能报告。

[0066] 在另一些实施例中，如图3所示，示出了自动化测试方法的框图。以游戏应用程序为例，产品的基本体验流程为，由蓝盾CI(Continuous Integration，持续集成)构建线300(游戏开发者开发好代码，需要产出安装包给用户，就是市面上的可安装应用程序，产出这个包的流水线，称之为构建线)产出测试包(游戏应用程序的安装包)后，执行包体下载和安装，然后由Python端驱动(指通过python代码驱动整个游戏的拉起、文件的交互和数据的解析过程)，进行自动化代码注入游戏客户端310，初始化自动化参数并push到手机游戏客户端。客户端启动后按照配置测试类型参数，自动化生成用例的json文件或者根据已有的用例文件初始化和自动化执行，比如测试主流程用例会给出步骤：进入换装、切换分类、执行换装、返回主场景，框架会按照用例一步步操作。自动化完成后，会主动关闭游戏进程，

Python驱动拉取游戏客户端生成的结果文件，进行解析，生成冒烟报告或者性能报告，可选地，使用AI(Artificial Intelligence，简称AI)机器人进行通知。

[0067] 相关技术中，手机客户端上的自动化测试，有GAutomator框架以及airtest框架。GAutomator框架主要是做UI自动化测试，通过在游戏包中打入一个SDK，在，一般使用Python脚本语言编写测试用例，向端口发送命令来控制引擎。这种方案可以通过直接修改编译白名单，使该SDK在编辑器环境下编译，用于做编辑器自动化测试。Airtest框架提供了跨平台的接口，包括安装应用、模拟输入、断言等。基于图像识别技术定位UI元素，无需嵌入任何代码即可进行自动化测试。测试脚本运行后可以自动生成详细的HTML测试报告，迅速定位失败的测试点。相关技术中的自动化测试方案大都在Gautomator或者Airtest上进行优化、拓展。主要有以下几个方向：整合驱动层：一种是将用例用可读性较好的几种语言进行编辑进行可视化的编辑，通过流程关键字控制。然后利用Gautomator或者Airtest进行自动化测试；整合执行层：利用GA-SDK在框架中调用代码的接口，整合游戏来调用工程中的代码逻辑，实现比较复杂的自动化行为。相关技术中，在虚幻引擎的编辑器上实现自动化测试方案有以下缺点：1、GAutomator方案主要UI自动化测试，对于较复杂的游戏，如FPS游戏，在局内需要控制角色移动，获取伤害等功能，单独通过UI操作难以实现；2、Airtest方案是基于图像识别来做自动化测试，但是对高渲染复杂度的游戏来说，单纯图像识别来做自动化比较局限，同时很难与游戏做交互来获取自动化用例中必要的某些具体数值与操作。在上述现有框架基础上进行优化拓展的行为，仍然受限于框架本身。比如基于GA的优化仍然面临着UI操作慢，等待时间长，无法进行复杂的游戏行为。相关技术中自动化的执行逻辑是外围驱动的。

[0068] 针对以上问题，本申请实施例提供的技术方案，一方面，剔除UI层面寻找控件，进行点击的自动化。通过在游戏工程内部实现用例调度行为，并完成数据结果的输出。由于是在内部实现了用例的调度，可自行封装接口，实现比较复杂的自动化测试行为、或者结合性能测试，获取深度性能数据等。本申请实施例提供的技术方案是分析游戏代码逻辑，了解内部的调度行为进行封装。进行点击的自动化是指不再在UI层面寻找控件并点击控件，而直接内部接口处理了。另一方面，在游戏工程面实现的自动化框架(本申请的自动化框架包括Python端驱动、Lua层接口封装以及控制执行的整个过程)，继承引擎框架特性，天然支持andorid系统、ios系统或者编辑器等。驱动层只需将游戏拉起，整个自动化由内置框架完成。框架不仅仅支持冒烟、同时支持性能测试等获取性能数据等行为，是一个集冒烟自动化、模式自动化、性能自动化与一体的自动化测试框架。本申请实施例在产品侧的应用主要是UE4游戏的三端自动化冒烟和性能测试。方案可以移植到其他的UE4游戏或其他项目中，可以在虚幻引擎的编辑器中直接使用。

[0069] 本申请实施例提供的技术方案是基于Lua的android、ios以及PC三端UE引擎自动化测试方案，实现了一套Python驱动的，集冒烟自动化、性能自动化及其他自动化与一体的自动化框架。游戏通过Unlua插件完成UE和Lua的交互，主要游戏代码在Lua层实现。框架梳理Lua层接口，并抽象出通用的接口，供自动化调用。Lua层利用单元测试的思想，与UE游戏Tick(延迟帧数)生命周期结合，在Tick中实现用例的驱动、调用和结果输出。用例存储在可配置的文件，可通过配置在初始化中写入。游戏一旦拉起即可通过框架的调用过程实现自动化测试。在游戏之外，通过Python控制游戏的拉起和结果的分析。整个过程减少了UI(用

户界面)层面等待的时间,基于接口设计执行,大大提高了执行成功率。

[0070] 请参考图4,其示出了本申请一个实施例提供的自动化测试方法的流程图。该方法各步骤的执行主体可以是图1所示方案实施环境中的终端设备10或者服务器20。在下文方法实施例中,为了便于描述,仅以各步骤的执行主体为“计算机设备”进行介绍说明。该方法可以包括如下几个步骤(410～450)中的至少一个步骤:

[0071] 步骤410,获取被测应用程序的自动化代码和初始化参数,自动化代码用于给被测应用程序提供自动化测试的环境,初始化参数用于控制自动化测试的执行。

[0072] 在一些实施例中,计算机设备在执行步骤410之前,先根据被测应用程序的项目代码文件安装被测应用程序的客户端。

[0073] 自动化代码:用于给被测应用程序提供自动化测试环境的代码。可选地,自动化代码可以拉取客户端,通常来说以游戏应用程序为例,需要用户手动点击登录操作,而通过自动化代码可以实现客户端的登录等拉起操作,无需用户手动点击。可选地,在接收到自动化代码之后,计算机设备具备了执行自动化测试的能力。可选地,计算机设备接收到的自动化代码除了可以用于拉起客户端之外,还可以用于自动化生成测试用例,并自动执行。进一步地,还可以将测试结果反馈给开发测试设备。

[0074] 初始化参数:用于控制自动化测试的执行的参数。可选地,初始化参数中的具体参数类型可以由开发测试人员自行设定。在一些实施例中,初始化参数用于告知计算机设备具体如何执行自动化测试,例如针对哪些功能进行自动化测试、如何生成测试用例等等。

[0075] 在一些实施例中,开发测试设备先将自动化代码发送给计算机设备,再将初始化参数发送给计算机设备。在另一些实施例中,开发测试设备将自动化代码以及初始化参数一起发送给计算机设备。可选地,自动化代码以及初始化参数均在开发测试设备上得到。可选地,开发测试设备使用Python脚本语言编写了自动化代码,并将该自动化代码以及初始化参数发送给计算机设备,驱使计算机设备执行自动化测试,实现了Python驱动。本申请实施例中,开发测试设备也是计算机设备,也即开发测试设备可以是终端设备或者服务器,本申请对此不作限定。可选地,自动化代码也需要调试或者变更,在自动化代码发生变化的情况下,重新从上述步骤410开始执行。

[0076] 可选地,在同时具备自动化代码以及初始化参数的情况下,计算机设备才可以开始执行后续步骤420～步骤450。

[0077] 步骤420,基于初始化参数,确定被测应用程序需要测试的至少一项功能。

[0078] 在一些实施例中,初始化参数中规定了计算机设备上需要测试的被测应用程序的功能。可选地,被测应用程序是游戏应用程序,假设该游戏应用程序的功能包括进入商城、换装、购买商品等等。可选地,先确定需要测试的功能,以需要测试的功能为换装为例,初始化参数中携带换装的标识信息,用于告知计算机设备所要测试的功能为换装。

[0079] 步骤430,对于每一项功能,对被测应用程序的项目代码文件中与功能相关的代码进行接口封装,得到功能对应的接口,功能对应的接口用于调用并执行功能相关的代码。

[0080] 项目文件代码是被测应用程序对应的实现代码。在一些实施例中,项目代码文件是通过计算机语言编写的代码。可选地,通过UE4来对被测应用程序进行代码开发调试,进而得到被测应用程序的项目代码文件。在一些实施例中,在UE4上通过Lua来编写被测应用程序的代码,实现热更。可选地,将调试好的项目代码文件发送给计算机设备,用于进行自

动化测试。可选地，在执行自动化测试之后，还需要继续对项目文件代码进行调试，在调试之后，继续执行上述步骤，重新进行自动化测试。

[0081] 可选地，被测应用程序可以根据用户需求实现很多的功能，因此，在项目代码文件会存在和不同功能对应的代码。可选地，该游戏应用程序的功能包括进入商城、换装、购买商品等等，则针对这些功能分别编写对应的代码，最终得到项目代码文件。可选地，将项目代码文件以功能为粒度进行划分，并进行接口封装，得到多个接口。可选地，通过调用接口，可以调用接口对应的代码。

[0082] 步骤440，生成用于对至少一项功能进行自动化测试的测试用例。

[0083] 测试用例根据开发者需求来编写的。可选地，测试用例是能够被计算机设备执行的代码文件。可选地，假设开发者需要验证换装这一功能，则通过在初始化参数写入测试需求，计算机设备可以根据需求，来生成关于换装的测试用例。并根据计算机设备的执行结果，得到测试用例的执行结果。测试用例的执行结果可能不符合不通过，也可能通过，如果通过则认为被测应用程序的项目代码文件的设计是符合开发者需求的，如果不通过则认为被测应用程序的项目代码文件的设计不符合开发者需求，需求重新对项目文件代码进行调试。

[0084] 在一些实施例中，根据自动化代码以及初始化参数，自动生成测试用例。可选地，在自动化代码中代码逻辑，使得计算机设备可以自行生成测试用例。

[0085] 步骤450，在执行测试用例的过程中，调用至少一项功能分别对应的接口以执行功能相关的代码，得到测试结果。

[0086] 在一些实施例中，仍以上述功能为换装为例，测试用例至少包括怎么进入换装，如何换装，以及换装之后的结果是什么。可选地，对于如何换装对应的项目代码文件中的代码进行封装，得到换装对应的接口。则在执行换装用例的时候，在步骤执行到如何换装，则调用换装对应的接口，找到对应的代码内容，执行代码，并继续往后执行，得到测试结果。

[0087] 在一些实施例中，在进行接口封装时，每个接口对应有标识信息。可选地，在生成测试用例时，将对应的接口的标识信息写入测试用例中，使得测试用例在执行时，可以根据具体的标识信息找到对应的接口，并调用具体的函数。

[0088] 可选地，测试结果是针对测试用例的结果。可选地，测试结果是测试通过还是不通过。可选地，测试结果还包括其他测试数据，例如性能测试数据、模式测试数据等等。可选地，性能测试数据可以是执行测试用例时的帧率、CPU(Central Processing Unit，中央处理器)的使用率等等。

[0089] 本申请实施例提供的技术方案，通过针对被测应用程序的自动化代码和初始化参数，可以拉起被测应用程序的客户端，并在客户端中生成自动化测试环境并控制开始执行自动化测试。一方面来说，通过初始化参数来明确被测应用程序需要测试的功能，并根据需要测试的功能来自动化生成测试用例。另一方面，对被测应用程序的项目代码文件进行接口封装，使得测试用例在执行过程中，可以直接调用接口，来快速地完成测试用例的执行。本申请实施例提供的技术方案，在接收到自动化代码和初始化参数之后，整个自动化测试流程均可以在同一个计算机设备中自行完成，无需其他开发测试设备的干预，因此，提升了自动化测试的效率。另外，由于针对需要测试的功能进行接口封装，在执行测试用例时，只需要快速调用对应的接口执行即可，缩短了测试所需要的时间。

[0090] 请参考图5,其示出了本申请另一个实施例提供的自动化测试方法的流程图。该方法各步骤的执行主体可以是图1所示方案实施环境中的终端设备10或者服务器20。在下文方法实施例中,为了便于描述,仅以各步骤的执行主体为“计算机设备”进行介绍说明。该方法可以包括如下几个步骤(410～450)中的至少一个步骤:

[0091] 步骤410,获取被测应用程序的自动化代码和初始化参数,自动化代码用于给被测应用程序提供自动化测试的环境,初始化参数用于控制自动化测试的执行。

[0092] 步骤421,初始化参数中包括第一类型参数,根据第一类型参数中的参数内容,确定被测应用程序需要测试的至少一项功能。

[0093] 在一些实施例中,第一类型参数用于确定需要测试的至少一项功能。在一些实施例中,第一类型参数是用于定义被测应用程序需要测试的功能的参数。可选地,被测应用程序的实际功能可能包括A功能、B功能、C功能、D功能。第一类型参数中可以定义需要测试的功能,例如需要测试的功能是A功能,则直接将A功能对应的标识信息作为第一类型参数。可选地,第一类型参数中定义的需要测试的功能可以为一个,也可以是多个,本申请对此不作限定。

[0094] 在一些实施例中,对不同的功能用不同的标识信息指代,在第一类型参数中定义需要测试的功能时仅需携带对应的标识信息即可。

[0095] 步骤430,对于每一项功能,对被测应用程序的项目代码文件中与功能相关的代码进行接口封装,得到功能对应的接口,功能对应的接口用于调用并执行功能相关的代码。

[0096] 步骤441,初始化参数中包括第二类型参数,在第二类型参数为第一数值的情况下,根据指定用例文件,生成测试用例,指定用例文件是执行过自动化测试并保存下来的用例文件。

[0097] 可选地,指定用例文件可以认为是在之前的自动化测试的过程中,被保存下来的文件。可选地,该指定用例文件被保存在本地或者云端,本申请对此不作限定。当保存在本地时,可以直接从存储器的对应位置找到对应的文件。当保存在云端时,可以向云端发送文件获取请求,请求获取该指定用例文件,从而获取该指定用例文件。

[0098] 在一些实施例中,第二类型参数用于指示测试用例的生成方式。本申请实施例对于第一数值以及第二数值的具体数值类型不作限定,可选地,第一数值是1,第二数值为0。当然,除去数值以外,还可以用其他的标识信息指代两种不同的测试用例的生成方式,对于标识信息的具体类型不作限定。

[0099] 在一些实施例中,步骤441包括步骤441-1～步骤441-2(图中未示出)中的至少一个步骤。

[0100] 步骤441-1,对指定用例文件进行解密及解析,得到解密及解析后的文件。

[0101] 在一些实施例中,当指定用例文件被保存时,并不是直接保存的,而是通过加密之后保存的,因此,当通过指定用例文件生成测试用例时,需要对该指定用例文件进行解密。在另一些实施例中,指定用例文件中不仅包括之前执行过自动化测试的测试用例,还包括该测试用例的执行结果等等,也即指定用例文件中并不完全是测试用例,还包括非测试用例的部分。因此,需要对该指定测试用例进行解析,确定出其中作为测试用例的部分。

[0102] 步骤441-2,对解密及解析后的文件进行用例初始化,得到测试用例。

[0103] 在另一些实施例中,需要将解密及解析后的文件对应的测试用例中针对的被测功

能与当前自动化参数指示的需要测试的功能进行比对，在二者一致的情况下，将解密及解析后的文件对应的测试用例作为步骤441-2中的测试用例。在二者不一致的情况下，根据当前自动化参数指示的需要测试的功能对解密及解析后的文件对应的测试用例中针对的被测功能进行适应性修改。例如，之前的测试用例是要求换第三套服装，而当前测试用例要求换第四套服装，则仅需对符合脏对应的编号标识进行适应性调整即可，而无需大幅度改动。

[0104] 步骤442，初始化参数中包括第二类型参数，在第二类型参数为第二数值的情况下，根据确定的需要测试的至少一项功能，生成所述测试用例。

[0105] 在一些实施例中，步骤442包括步骤442-1～步骤442-2(图中未示出)中的至少一个步骤。

[0106] 步骤442-1，确定需要测试的至少一项功能分别对应的接口。

[0107] 可选地，根据需要测试的至少一项功能，确定其对应的接口。例如，仍以换装为例，如果需要测试的功能为换装，则找到换装功能对应的接口。

[0108] 步骤442-2，据自动化代码中携带的用例格式信息，基于需要测试的至少一项功能分别对应的接口，生成测试用例。

[0109] 用例格式信息是用于帮助生成用例的信息。在一些实施例中，用例格式信息是在自动化代码中自行设定的，计算机设备在接收到自动化代码之后，只需要根据其中携带的用例格式信息，来生成测试用例。在一些实施例中，用例格式信息中规定了不同功能对应的用例的输入以及输出信息、执行函数、执行逻辑等等。

[0110] 步骤450，在执行测试用例的过程中，调用至少一项功能分别对应的接口以执行功能相关的代码，得到测试结果。

[0111] 如图6所示，600示出了测试用例生成以及执行的过程，在主Tick(这里可以理解被测应用程序的主要运行逻辑或者主要实现的功能，以换装游戏为例，主tick也就换装过程)内，首先，判断是否需要自动化生成测试用例，如果不需要自动化生成，则对指定用文件进行初始化，得到测试用例。如果需要自动化生成，则自动化生成测试用例。在用例执行过程中，判断是否执行完成，如果执行完成则结束，如果执行未完成，则继续执行下一步，并写入用例结果。

[0112] 本申请实施例提供的技术方案，通过在自动化参数中设定第一类型参数，可以指示计算机设备去针对指定的用例来进行自动化测试，也即设定测试目标，而由计算机设备根据目标来进行测试，可以在一定程度上简化测试流程，避免无异议的多余测试。

[0113] 另外，通过在自动化参数中设定第二类型参数，可以指示计算机设备生成测试用例的方法，一方面可以根据已有的指定用例文件来进行用例初始化，也即无需完整重新生成新的测试用例，在仅在指定用例文件的基础上进行初始化的简单修改，即可得到测试用例，因此可以简化测试用例的生成步骤，减少设备的处理开销。另一方面，自动化生成用例，根据测试需求实时生成测试用例，可以进一步保证生成的测试用例的准确性以及灵活性。

[0114] 请参考图7，其示出了本申请另一个实施例提供的自动化测试方法的流程图。该方法各步骤的执行主体可以是图1所示方案实施环境中的终端设备10或者服务器20。在下文方法实施例中，为了便于描述，仅以各步骤的执行主体为“计算机设备”进行介绍说明。该方法可以包括如下几个步骤(410～470)中的至少一个步骤：

[0115] 步骤410，获取被测应用程序的自动化代码和初始化参数，自动化代码用于给被测

应用程序提供自动化测试的环境,初始化参数用于控制自动化测试的执行。

[0116] 步骤420,基于初始化参数,确定被测应用程序需要测试的至少一项功能。

[0117] 步骤431,将被测应用程序的项目代码文件中的第一代码文件中与第一类型功能相关的代码进行接口封装,得到功能对应的接口。

[0118] 可选地,第一类型功能以及第二类型功能是相同或者不同的功能。可选地,第一类型功能或者第二类型功能均认为是被测应用程序中的通用功能。具体通用功能的种类可以由开发测试人员提前设定好,本申请对此不作限定。

[0119] 步骤432,将被测应用程序的项目代码文件中的第二代码文件中与第二类型功能相关的代码进行接口封装,得到功能对应的接口,第一类型功能和第二类型功能是被测应用程序所提供的不同的功能。

[0120] 可选地,项目代码文件包括第一代码文件和第二代码文件,其中,第一代码文件是使用第一编程语言编写的代码文件,第二代码文件是使用第二编程语言编写的代码文件,第一编程语言和第二编程语言不同,第一代码文件和第二代码文件分别实现被测应用程序的不同功能。可选地,第一代码文件和第二代码文件可以互相调用。

[0121] 在一些实施例中,第一代码文件是利用Lua对应的编程语言而编写的代码。第二代码文件是在虚幻引擎中利用C语言编写的。可选地,通过插件连接第一代码文件以及第二代码文件,从而使得第一代码文件以及第二代码文件构成项目文件代码。可选地,插件包括但不限于Unlua、slua、ulua、xlua等等。本申请实施例中以Unlua来示例。当然,本申请实施例对于具体的编程语言的类型不作限定。可选地,对于被测应用程序中通用的且不会轻易改变的功能在虚幻引擎中利用C语言编写得到第一代码文件,对于被测应用程序中不通用的且会轻易改变的功能在Lua中得到第二代码文件。这是考虑到Lua可热更的特性,利用Lua对于被测应用程序中不通用的且会轻易改变的功能进行代码编写,对于被测应用程序的开发调试较为友好。此处具体解释下代码调用原理:Unlua是其中一种游戏热更框架,C++调用Lua实际上是:由C++先把数据放入栈中,由Lua去栈中取数据,然后返回数据对应的值到栈顶,再由栈顶返回C++。Lua调C++也一样:先编写自己的C模块,然后注册函数到Lua解释器中,然后由Lua去调用这个模块的函数。Unlua在这个基础上进行了进一步的封装,使得可以更方便的使用。

[0122] 步骤440,生成用于对至少一项功能进行自动化测试的测试用例。

[0123] 步骤460,在测试用例执行之前,基于自动化代码,确定用例执行控制逻辑,用例执行控制逻辑包括以下至少之一:用例依赖情况信息、执行函数信息、跳转信息、主流程验证信息、重试指示信息、保存方式信息。

[0124] 在一些实施例中,用例执行控制逻辑是在存在测试用例的情况下,执行测试用例的逻辑。其中,用例依赖情况信息指示了测试用例之间的依赖关系,执行函数信息指示了执行测试用例所需要的函数,跳转信息指示了执行步骤或者执行用例之间的跳转关系,主流程验证信息指示了是否需要主流程(主Tick)验证。重试指示信息指示了用例执行失败之后是否需要重新执行,保存方式信息指示了如何保存用例结果(以截图还是快照等形式)。

[0125] 步骤470,基于用例执行控制逻辑,执行测试用例。

[0126] 在一些实施例中,步骤470包括步骤470-1～步骤470-3(图中未示出)中的至少一个步骤。

[0127] 步骤470-1,在执行测试用例之前,根据用例执行控制逻辑包括的用例依赖情况信息所指示的测试用例对应的依赖用例的执行情况,确定是否执行测试用例。

[0128] 在一些实施例中,存在一些测试用例之间存在先后关系,必须前一个测试用例执行完成之后才可以执行下一个测试用例,当前一个测试用例执行失败的情况下,下一个测试用例无法正常执行。因此,在执行测试用例时,首先需要根据用例依赖情况信息确定依赖用例是否执行完成。

[0129] 步骤470-2,在执行测试用例的过程中,根据用例执行控制逻辑包括的跳转信息,在测试用例中的第一步骤执行成功的情况下,从第一步骤跳转至第二步骤开始执行,第一步骤和第二步骤之间的跳转关系存储在跳转信息中,第二步骤是测试用例中除去第一步骤以外的其他步骤,或其他测试用例中的步骤。

[0130] 在一些实施例中,在测试用例中的第一步骤执行成功的情况下,根据跳转信息判断是否需要跳转,如果需要跳转,则从第一步骤跳转至第二步骤开始执行,第二步骤并非是第一步骤的下一个步骤。如果不需要跳转,则从第一步骤的下一个步骤开始执行。

[0131] 步骤470-3,在测试用例执行失败的情况下,根据用例执行控制逻辑包括的重试指示信息,对测试用例进行重新测试。

[0132] 在一些实施例中,重试指示信息还指示重试次数。在满足重试次数之后,如果测试用例还是执行失败,则不再继续重新测试。

[0133] 可选地,图8的800示出了执行测试用例的示意图。可选地,在执行一个测试用例时,首先需要判断其依赖的用例是否执行成功,在依赖的用例执行失败的情况下,根据跳转信息,进行跳转,例如跳转到最后。在依赖用例执行成功的情况下,执行测试用例的步骤,并进行主流程验证,如果验证成功,则根据跳转信息确定是否跳转,如果验证失败,则判断是否需要重试,接着进行截图保存测试结果。

[0134] 在一些实施例中,执行测试用例得到测试结果。可选地,测试结果中包括第一类型数据和第二类型数据,第一类型数据反映测试用例是否通过,第二类型数据表征测试用例在执行过程中的性能数据;第一类型数据用于生成冒烟报告,第二类型数据用于生成性能报告。

[0135] 在一些实施例中,第二类型数据包括但不限于测试用例执行过程中的帧率、CPU温度、CPU使用占比等等。

[0136] 在一些实施例,包括第一类型数据以及第二类型数据的测试结果发送给开发测试设备,开发测试设备接收到测试结果之后,根据第一类型数据生成冒烟报告,根据第二类型数据生成性能报告。如图9的900所示,是一份冒烟报告的示意图,其中示出了哪些测试用例通过,哪些测试用例未通过,并且给出了具体的未通过的原因。

[0137] 在一些实施例中,冒烟简单确认游戏包是否具备可测条件,比如游戏登录、进大厅、匹配、局内等主流程是否通过,这个过程可以称之为冒烟。冒烟自动化就是使用自动化程序代替人工做。性能这里指游戏客户端性能,性能自动化是指将性能测试过程、性能数据获取等自动化过程。模式自动化类似冒烟、性能是一种业务的逻辑。比如A游戏有匹配模式、有开房间模式,有1v1、有5v5等、这些都可以在框架上来做,以实现模式自动化。因此,本申请实施例提供的技术方案,可以实现冒烟自动化、性能自动化以及模式自动化。

[0138] 本申请实施例提供的技术方案,通过设定用例执行控制逻辑,保证了测试用例的

平稳运行,有利于提高自动化测试的执行成功率。在另一些方面,增加重试和跳转机制,提升自动化测试的准确度并缩短了自动化测试的时间。

[0139] 另外,本申请实施例在进行接口封装时,根据项目代码文件中对应的不同编程语言编写的代码分别进行封装,并且是根据指定的功能来封装的,因此,封装出来的接口是后续自动化测试中需要用到的接口,在后续执行自动化测试过程中,直接通过接口调用代码来执行即可,有利于提升自动化测试的执行成功率,并且提升了自动化测试的效率。

[0140] 请参考图10,其示出了本申请一个实施例提供的自动化测试框架的示意图。

[0141] 本申请实施例提供的自动化框架包括Python驱动层1000以及客户端执行层1010。

[0142] 可选地,开发测试设备仅需要设计一套自动化代码,也即指测试人员只需要用Python编写一套驱动安卓和ios代码即可。通过Python驱动层1000可以动态修改发送给游戏客户端的自动化代码。Lua自动化代码本身不污染游戏客户端,可独立放置,由Python驱动层push到文件固定目录下,完成游戏客户端的动态修改。可选地,客户端包括三端,分别是运行在android、ios、编辑器环境上的。编辑器自启动自动化,安卓和ios则在Python端通过Appium驱动完成,也可以理解为通过Appium来和安卓、ios的模拟器或者真机建立连接,在建立连接之后,可以由开发测试设备向模拟器或者真机发送自动化代码,实现Python驱动自动化测试。

[0143] 客户端执行层1010包括Lua控制层、Unlua插件以及UE4引擎。这里的UE4引擎可以认为是通过UE4引擎开发的项目文件代码对应的发送给终端的安装包中的关于UE4引擎的代码,认为是UE4引擎。客户端执行层1010执行了上述自动化测试方法的步骤。

[0144] Lua控制层:UE4原生支持Lua,Lua由于生来可热更,写完不用等编译,在线编程(一边运行一边编写),受到一众开发者的青睐。Lua控制层利用Unlua架起Lua和UE的桥梁。主要实现:Lua接口封装以及Lua流程控制。

[0145] Lua接口封装:放弃了相关技术中外层UI点击控制,改用接口的封装调用的形式进行自动化。通过unlua特性,可随意调用UE4函数,封装成Lua控制层可调用的接口。如图11的子图a所示,其示出了自动化代码的代码示意图。如图11的子图b所示,其示出了接口的代码示意图。

[0146] Lua流程控制:封装好各种各样的接口后,Lua控制层在主Tick内进行流程的控制,Tick中首先根据传入的初始化参数决定是否需要自动化生成执行的用例,如果需要则调用相关接口,生成自动化用例,如果否则,则根据指定的用例文件进行case初始化,用例文件示例。初始化相关用例后,则需要控制整体执行逻辑。根据配置的用例的依赖情况、执行函数、是否需要主流程验证、是否重试、是否截图等参数。如图12的子图a所示,其示出了生成的测试用例的代码示意图。如图12的子图b所示,其示出了指定用例文件的代码示意图。

[0147] 本申请实施例提供了一种基于UE4-Lua的多端自动化测试方案,通过Unlua插件完成UE和Lua的交互,主要游戏代码在Lua控制层实现。框架梳理Lua控制层接口,并抽象出通用的接口,供自动化调用。Lua控制层利用单元测试的思想,与UE游戏Tick生命周期结合,在Tick中实现用例的驱动、调用和结果输出。用例存储在可配置的文件,可通过配置在初始化中写入。游戏一旦拉起即可通过框架的调用过程实现自动化测试。在游戏之外,通过python控制游戏的拉起和结果的分析。整个过程不污染项目代码,同时自动化测试减少了UI层面等待的时间,基于接口设计执行,大大提高了执行成功率。

[0148] 下述为本申请装置实施例，可以用于执行本申请方法实施例。对于本申请装置实施例中未披露的细节，请参照本申请方法实施例。

[0149] 请参考图13，其示出了本申请一个实施例提供的自动化测试装置的框图。该装置1300可以包括：获取模块1310、功能确定模块1320、接口封装模块1330、用例生成模块1340以及用例执行模块1350。

[0150] 所述获取模块1310，用于获取被测应用程序的自动化代码和初始化参数，所述自动化代码用于给所述被测应用程序提供自动化测试的环境，所述初始化参数用于控制所述自动化测试的执行。

[0151] 所述功能确定模块1320，用于基于所述初始化参数，确定所述被测应用程序需要测试的至少一项功能。

[0152] 所述接口封装模块1330，用于对于每一项功能，对所述被测应用程序的项目代码文件中与所述功能相关的代码进行接口封装，得到所述功能对应的接口，所述功能对应的接口用于调用并执行所述功能相关的代码。

[0153] 所述用例生成模块1340，用于生成用于对所述至少一项功能进行自动化测试的测试用例。

[0154] 所述用例执行模块1350，用于在执行所述测试用例的过程中，调用所述至少一项功能分别对应的接口以执行所述功能相关的代码，得到测试结果。

[0155] 在一些实施例中，所述初始化参数中包括第一类型参数，所述第一类型参数用于确定所述需要测试的至少一项功能。

[0156] 所述功能确定模块1320，用于根据所述第一类型参数中的参数内容，确定所述被测应用程序需要测试的至少一项功能。

[0157] 在一些实施例中，所述初始化参数中包括第二类型参数，所述第二类型参数用于指示测试用例的生成方式。

[0158] 在一些实施例中，如图14所示，所述用例生成模块1340包括第一生成单元1341以及第二生成单元1342。

[0159] 所述第一生成单元1341，用于在所述第二类型参数为第一数值的情况下，根据指定用例文件，生成所述测试用例，所述指定用例文件是执行过自动化测试并保存下来的用例文件；或者，第二生成单元1342，用于在所述第二类型参数为第二数值的情况下，根据确定的所述需要测试的至少一项功能，生成所述测试用例；其中，所述第一数值与所述第二数值不同。

[0160] 在一些实施例中，所述第一生成单元1341，用于对所述指定用例文件进行解密及解析，得到解密及解析后的文件。

[0161] 所述第一生成单元1341，还用于对所述解密及解析后的文件进行用例初始化，得到所述测试用例。

[0162] 在一些实施例中，第二生成单元1342，用于确定所述需要测试的至少一项功能分别对应的接口。

[0163] 第二生成单元1342，还用于根据所述自动化代码中携带的用例格式信息，基于所述需要测试的至少一项功能分别对应的接口，生成所述测试用例。

[0164] 在一些实施例中，所述项目代码文件包括第一代码文件和第二代码文件，其中，所

述第一代码文件是使用第一编程语言编写的代码文件,所述第二代码文件是使用第二编程语言编写的代码文件,所述第一编程语言和所述第二编程语言不同,所述第一代码文件和所述第二代码文件分别实现所述被测应用程序的不同功能。

[0165] 所述接口封装模块1330,用于将所述第一代码文件中与第一类型功能相关的代码进行接口封装,得到所述功能对应的接口。

[0166] 所述接口封装模块1330,还用于将所述第二代码文件中与第二类型功能相关的代码进行接口封装,得到所述功能对应的接口,所述第一类型功能和所述第二类型功能是所述被测应用程序所提供的不同的功能。

[0167] 在一些实施例中,所述测试用例的数量为至少一个,所述装置还包括逻辑确定模块1360。

[0168] 所述逻辑确定模块1360,用于在所述测试用例执行之前,基于所述自动化代码,确定用例执行控制逻辑,所述用例执行控制逻辑包括以下至少之一:用例依赖情况信息、执行函数信息、跳转信息、主流程验证信息、重试指示信息、保存方式信息。

[0169] 所述用例执行模块1350,还用于基于所述用例执行控制逻辑,执行所述测试用例。

[0170] 在一些实施例中,所述用例执行模块1350,用于在执行所述测试用例之前,根据所述用例执行控制逻辑包括的所述用例依赖情况信息所指示的所述测试用例对应的依赖用例的执行情况,确定是否执行所述测试用例。

[0171] 所述用例执行模块1350,还用于在执行所述测试用例的过程中,根据所述用例执行控制逻辑包括的所述跳转信息,在所述测试用例中的第一步骤执行成功的情况下,从所述第一步骤跳转至第二步骤开始执行,所述第一步骤和所述第二步骤之间的跳转关系存储在所述跳转信息中,所述第二步骤是所述测试用例中除去所述第一步骤以外的其他步骤,或其他测试用例中的步骤。

[0172] 所述用例执行模块1350,还用于在所述测试用例执行失败的情况下,根据所述用例执行控制逻辑包括的所述重试指示信息,对所述测试用例进行重新测试。

[0173] 在一些实施例中,所述测试结果中包括第一类型数据和第二类型数据,所述第一类型数据反映所述测试用例是否通过,所述第二类型数据表征所述测试用例在执行过程中的性能数据;所述第一类型数据用于生成冒烟报告,所述第二类型数据用于生成性能报告。

[0174] 需要说明的是,上述实施例提供的装置,在实现其功能时,仅以上述各功能模块的划分进行举例说明,实际应用中,可以根据需要而将上述功能分配由不同的功能模块完成,即将设备的内部结构划分成不同的功能模块,以完成以上描述的全部或者部分功能。另外,上述实施例提供的装置与方法实施例属于同一构思,其具体实现过程详见方法实施例,这里不再赘述。

[0175] 图15示出了本申请另一个示例性实施例提供的计算机设备的结构框图。

[0176] 通常,计算机设备1500包括有:处理器1501和存储器1502。

[0177] 处理器1501可以包括一个或多个处理核心,比如4核心处理器、15核心处理器等。处理器1501可以采用DSP(Digital Signal Processing,数字信号处理)、FPGA(Field Programmable Gate Array,现场可编程门阵列)、PLA(Programmable Logic Array,可编程逻辑阵列)中的至少一种硬件形式来实现。处理器1501也可以包括主处理器和协处理器,主处理器是用于对在唤醒状态下的数据进行处理的处理器,也称CPU;协处理器是用于对在待

机状态下的数据进行处理的低功耗处理器。在一些实施例中，处理器1501可以在集成有GPU(Graphics Processing Unit，图像处理器)，GPU用于负责显示屏所需要显示的内容的渲染和绘制。一些实施例中，处理器1501还可以包括AI处理器，该AI处理器用于处理有关机器学习的计算操作。

[0178] 存储器1502可以包括一个或多个计算机可读存储介质，该计算机可读存储介质可以是有形的和非暂态的。存储器1502还可包括高速随机存取存储器，以及非易失性存储器，比如一个或多个磁盘存储设备、闪存存储设备。在一些实施例中，存储器1502中的非暂态的计算机可读存储介质存储有计算机程序，该计算机程序由处理器1501加载并执行以实现上述自动化测试方法。

[0179] 本领域技术人员可以理解，图15中示出的结构并不构成对计算机设备1500的限定，可以包括比图示更多或更少的组件，或者组合某些组件，或者采用不同的组件布置。

[0180] 在示例性实施例中，还提供了一种计算机可读存储介质，存储介质中存储有计算机程序，计算机程序在被处理器执行时以实现上述方法。

[0181] 可选地，该计算机可读存储介质可以包括：ROM(Read-Only Memory，只读存储器)、RAM(Random Access Memory，随机存取存储器)、SSD(Solid State Drives，固态硬盘)或光盘等。其中，随机存取存储器可以包括ReRAM(Resistance Random Access Memory，电阻式随机存取存储器)和DRAM(Dynamic Random Access Memory，动态随机存取存储器)。

[0182] 在示例性实施例中，还提供了一种计算机程序产品，计算机程序产品包括计算机程序，计算机程序存储在计算机可读存储介质中。计算机设备的处理器从计算机可读存储介质中读取计算机程序，处理器执行计算机程序，使得计算机设备执行上述自动化测试方法。

[0183] 应当理解的是，在本文中提及的“多个”是指两个或两个以上。“和/或”，描述关联对象的关联关系，表示可以存在三种关系，例如，A和/或B，可以表示：单独存在A，同时存在A和B，单独存在B这三种情况。字符“/”一般表示前后关联对象是一种“或”的关系。另外，本文中描述的步骤编号，仅示例性示出了步骤间的一种可能的执行先后顺序，在一些其它实施例中，上述步骤也可以不按照编号顺序来执行，如两个不同编号的步骤同时执行，或者两个不同编号的步骤按照与图示相反的顺序执行，本申请实施例对此不作限定。

[0184] 以上仅为本申请的示例性实施例，并不用以限制本申请，凡在本申请的精神和原则之内，所作的任何修改、等同替换、改进等，均应包含在本申请的保护范围之内。

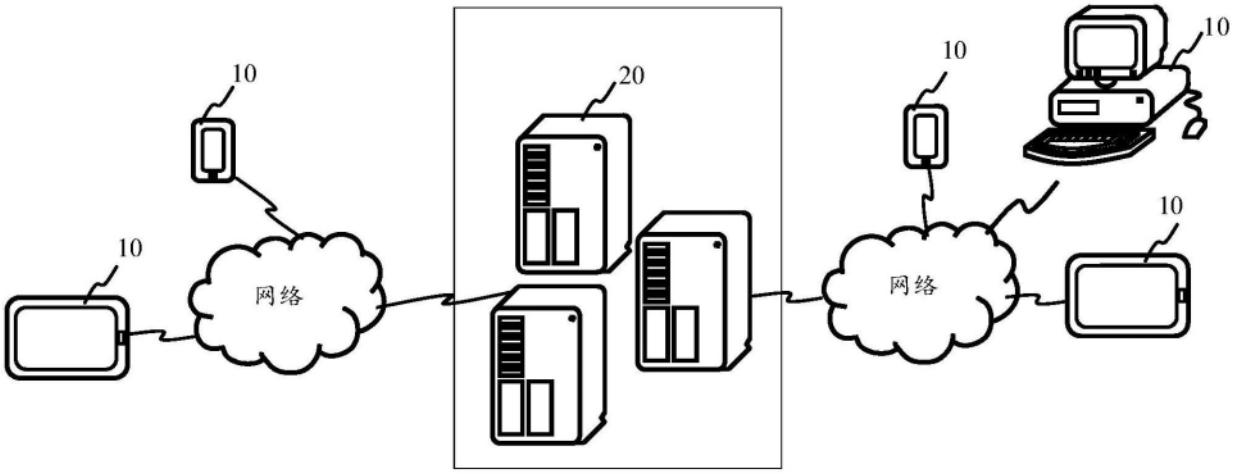


图1

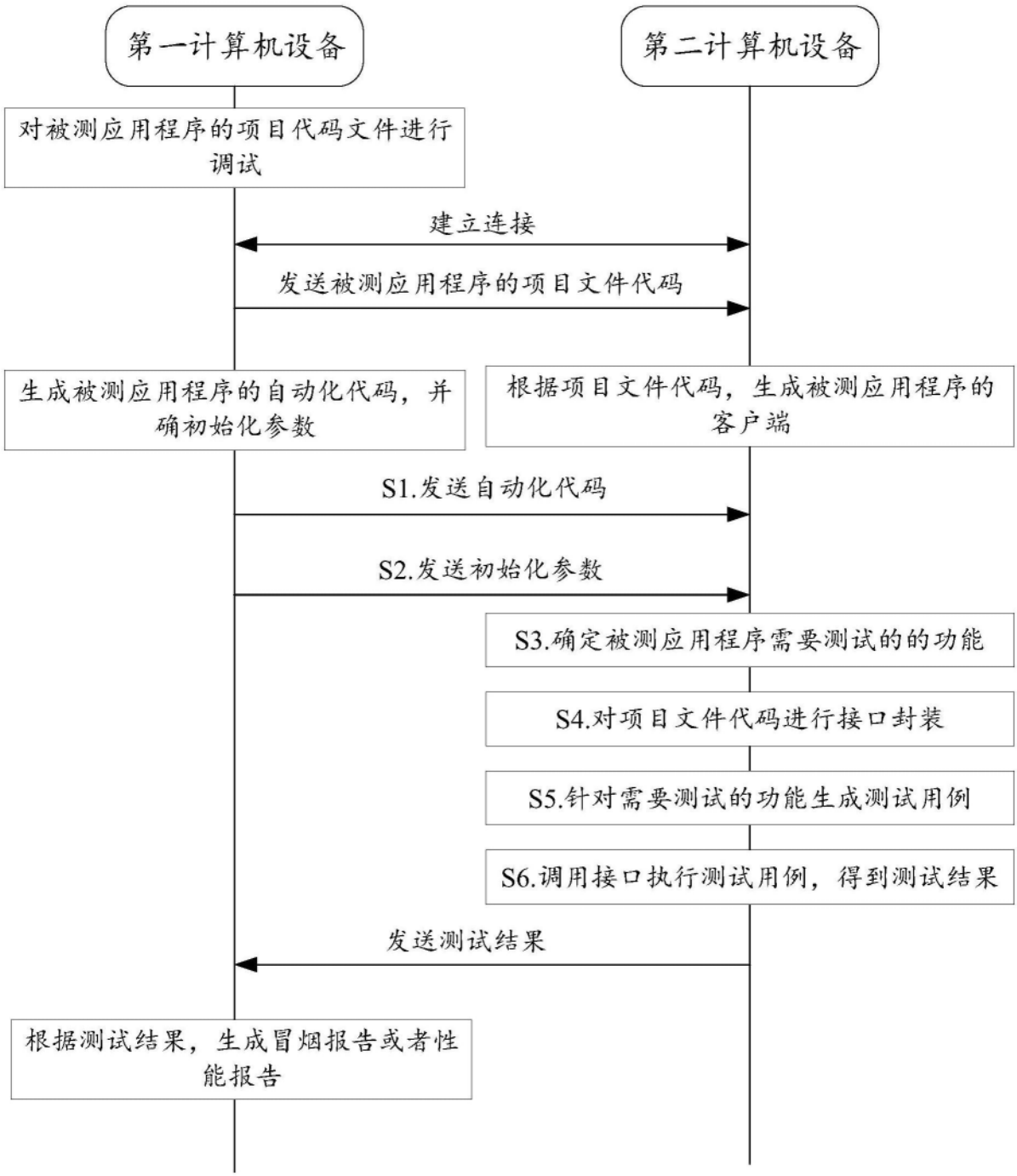

第一计算机设备
第二计算机设备
对被测应用程序的项目代码文件进行调试
建立连接
发送被测应用程序的项目文件代码
生成被测应用程序的自动化代码，并确初始化参数
根据项目文件代码，生成被测应用程序的客户端
S1.发送自动化代码
S2.发送初始化参数
S3.确定被测应用程序需要测试的的功能
S4.对项目文件代码进行接口封装
S5.针对需要测试的功能生成测试用例
S6.调用接口执行测试用例，得到测试结果
发送测试结果
根据测试结果，生成冒烟报告或者性能报告

图2

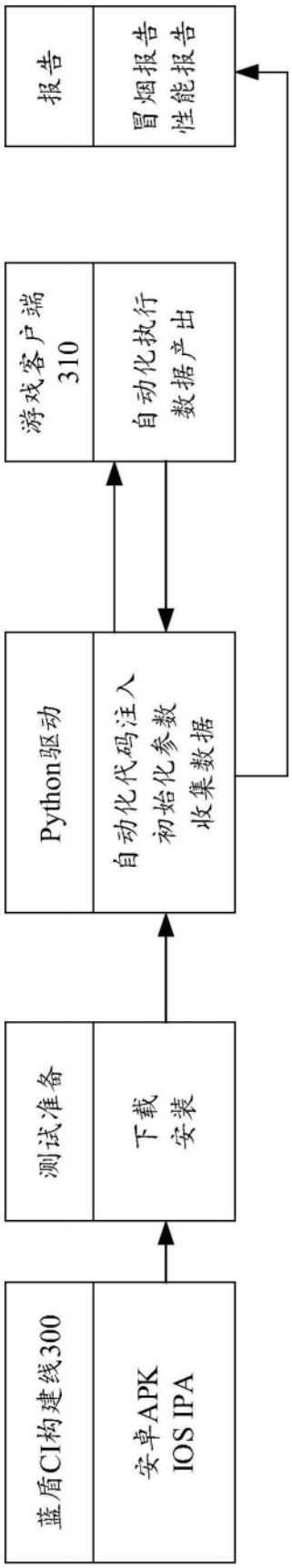

蓝盾CI构建线300
安卓APK
IOS IPA
测试准备
下载
安装
Python驱动
自动化代码注入
初始化参数
收集数据
游戏客户端
310
自动化执行
数据产出
报告
冒烟报告
性能报告

图3

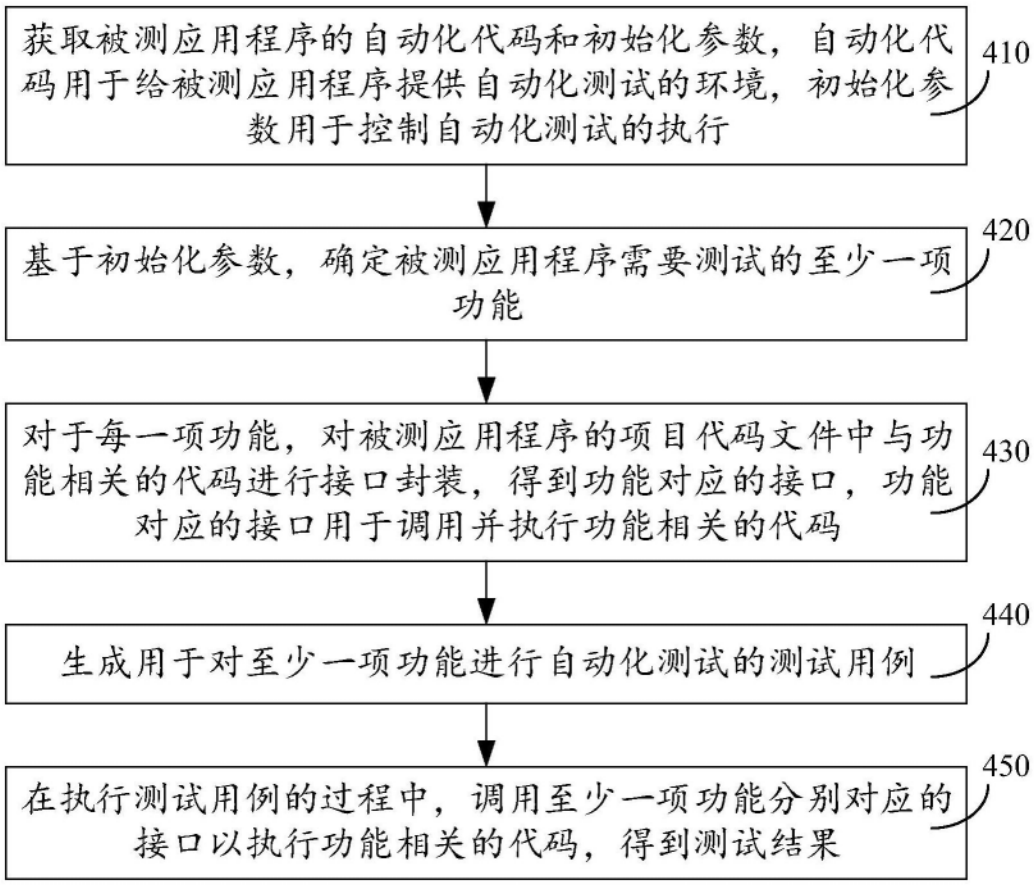

获取被测应用程序的自动化代码和初始化参数，自动化代码用于给被测应用程序提供自动化测试的环境，初始化参数用于控制自动化测试的执行
410
基于初始化参数，确定被测应用程序需要测试的至少一项功能
420
对于每一项功能，对被测应用程序的项目代码文件中与功能相关的代码进行接口封装，得到功能对应的接口，功能对应的接口用于调用并执行功能相关的代码
430
生成用于对至少一项功能进行自动化测试的测试用例
440
在执行测试用例的过程中，调用至少一项功能分别对应的接口以执行功能相关的代码，得到测试结果
450

图4

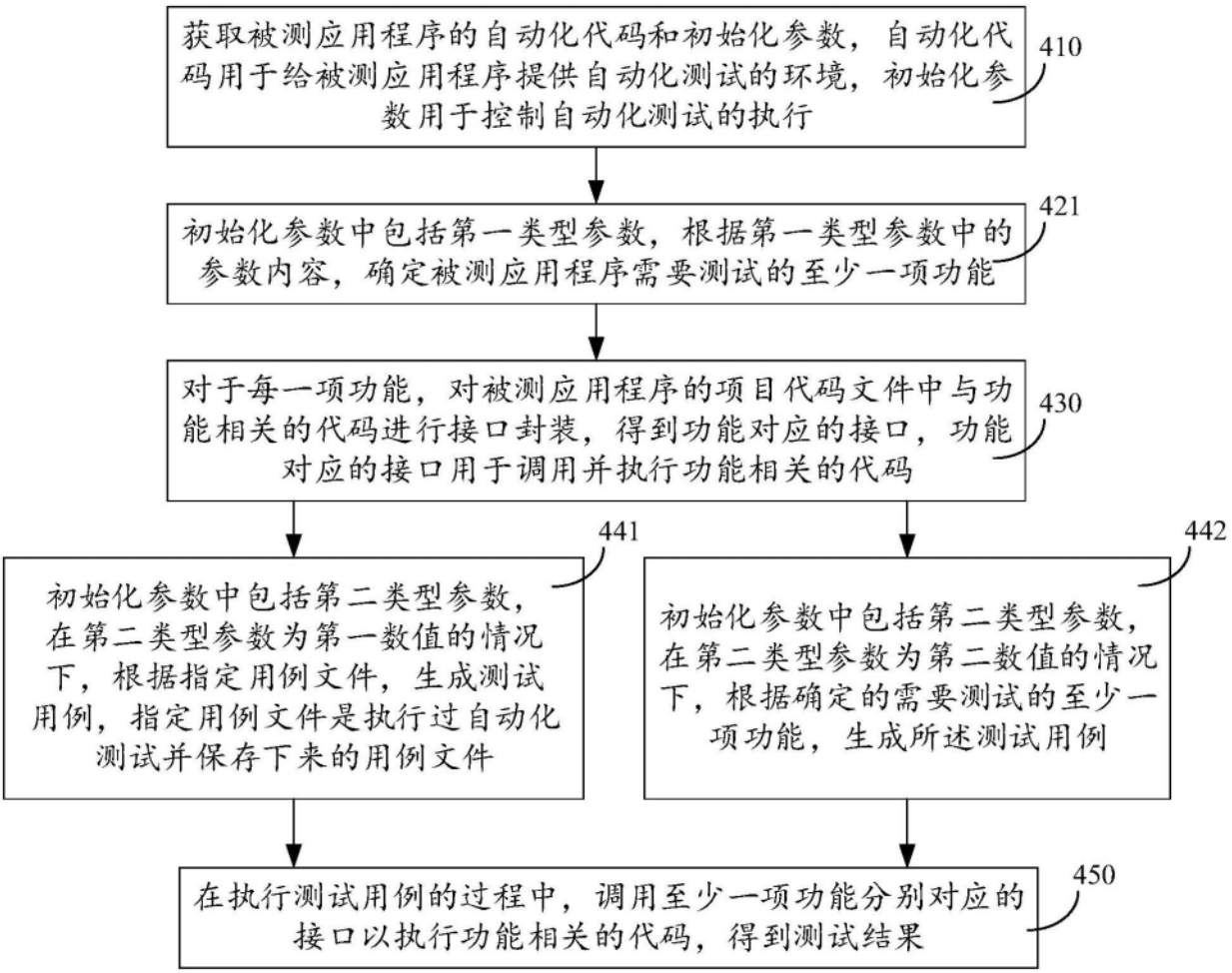

获取被测应用程序的自动化代码和初始化参数，自动化代码用于给被测应用程序提供自动化测试的环境，初始化参数用于控制自动化测试的执行
410
初始化参数中包括第一类型参数，根据第一类型参数中的参数内容，确定被测应用程序需要测试的至少一项功能
421
对于每一项功能，对被测应用程序的项目代码文件中与功能相关的代码进行接口封装，得到功能对应的接口，功能对应的接口用于调用并执行功能相关的代码
430
441
初始化参数中包括第二类型参数，在第二类型参数为第一数值的情况下，根据指定用例文件，生成测试用例，指定用例文件是执行过自动化测试并保存下来的用例文件
442
初始化参数中包括第二类型参数，在第二类型参数为第二数值的情况下，根据确定的需要测试的至少一项功能，生成所述测试用例
在执行测试用例的过程中，调用至少一项功能分别对应的接口以执行功能相关的代码，得到测试结果
450

图5

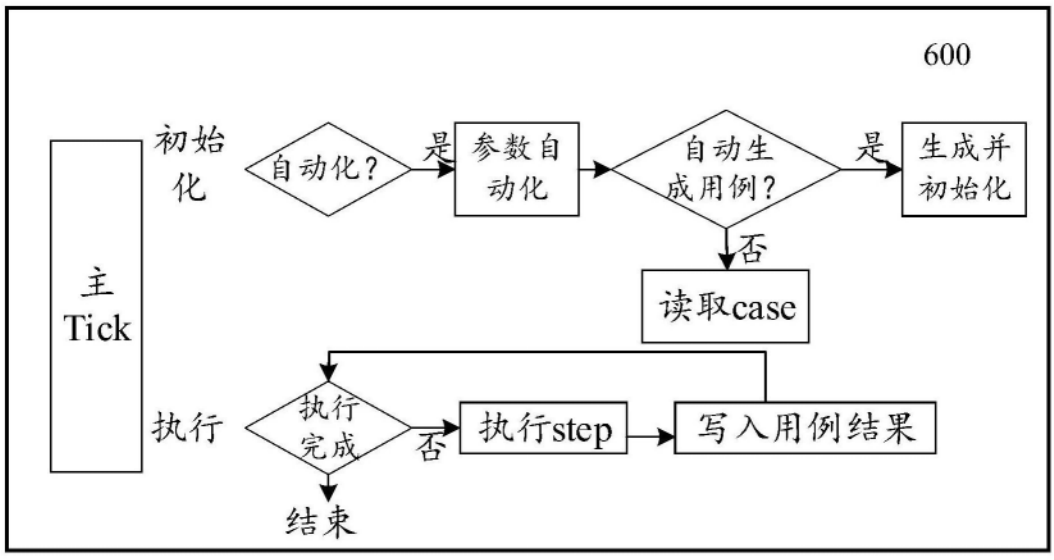

600
主
Tick
初始化
自动化?
是
参数自动化
自动生成用例?
是
生成并初始化
否
读取case
执行
执行完成
否
执行step
写入用例结果
结束

图6

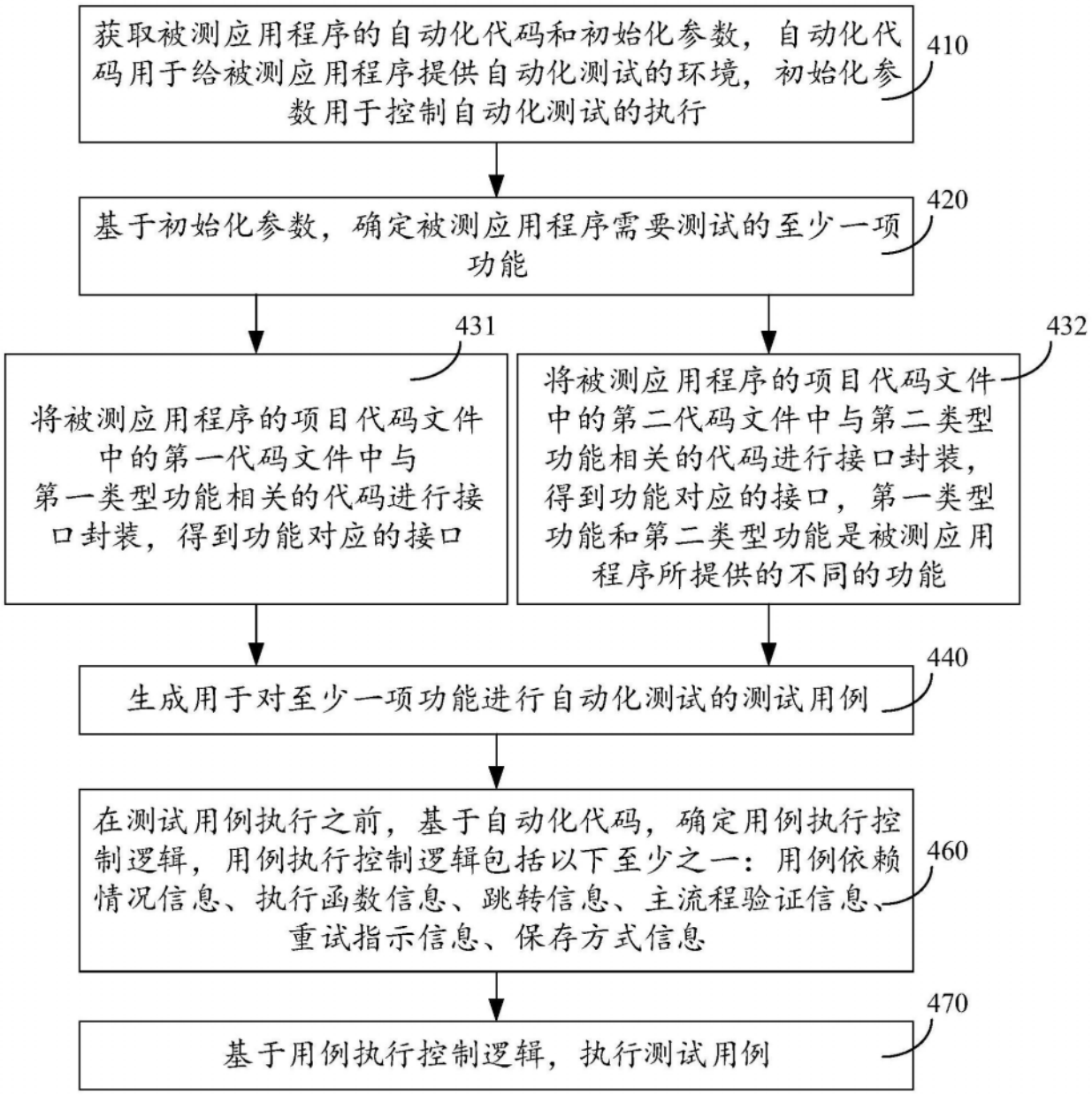

获取被测应用程序的自动化代码和初始化参数，自动化代码用于给被测应用程序提供自动化测试的环境，初始化参数用于控制自动化测试的执行
410
基于初始化参数，确定被测应用程序需要测试的至少一项功能
420
431
将被测应用程序的项目代码文件中的第一代码文件中与第一类型功能相关的代码进行接口封装，得到功能对应的接口
432
将被测应用程序的项目代码文件中的第二代码文件中与第二类型功能相关的代码进行接口封装，得到功能对应的接口，第一类型功能和第二类型功能是被测应用程序所提供的不同的功能
440
生成用于对至少一项功能进行自动化测试的测试用例
在测试用例执行之前，基于自动化代码，确定用例执行控制逻辑，用例执行控制逻辑包括以下至少之一：用例依赖情况信息、执行函数信息、跳转信息、主流程验证信息、重试指示信息、保存方式信息
460
470
基于用例执行控制逻辑，执行测试用例

图7

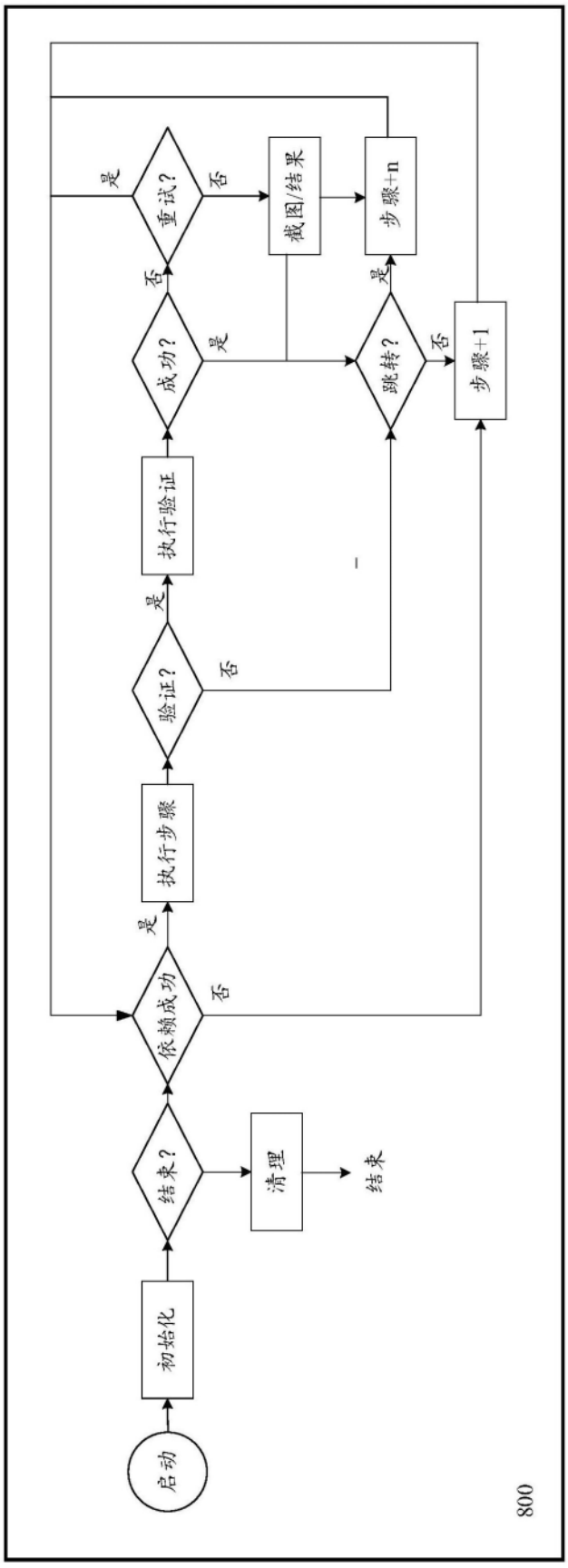


图8

Android端自动化测试用例执行 900 执行地址：xxx 用例执行概况 成功用例数目：24 失败用例数目：2 跳过用例数：0 进入商城：通过 切换至背包：通过 背包穿搭方案：通过 背包切换不同模块：通过 背包筛选：通过 切换至换装：通过 商城筛选：通过 确认筛选：通过 -确认筛选：不通过 原因：第4个商品和第5个商品没有按照价格类型排序 商品详情页：不通过 -打开商品扩展菜单：不通过 原因：xx没有打开 连续换装：通过 商品购买：通过 装扮推荐：通过
失败用例及日志
下载安装包

图9

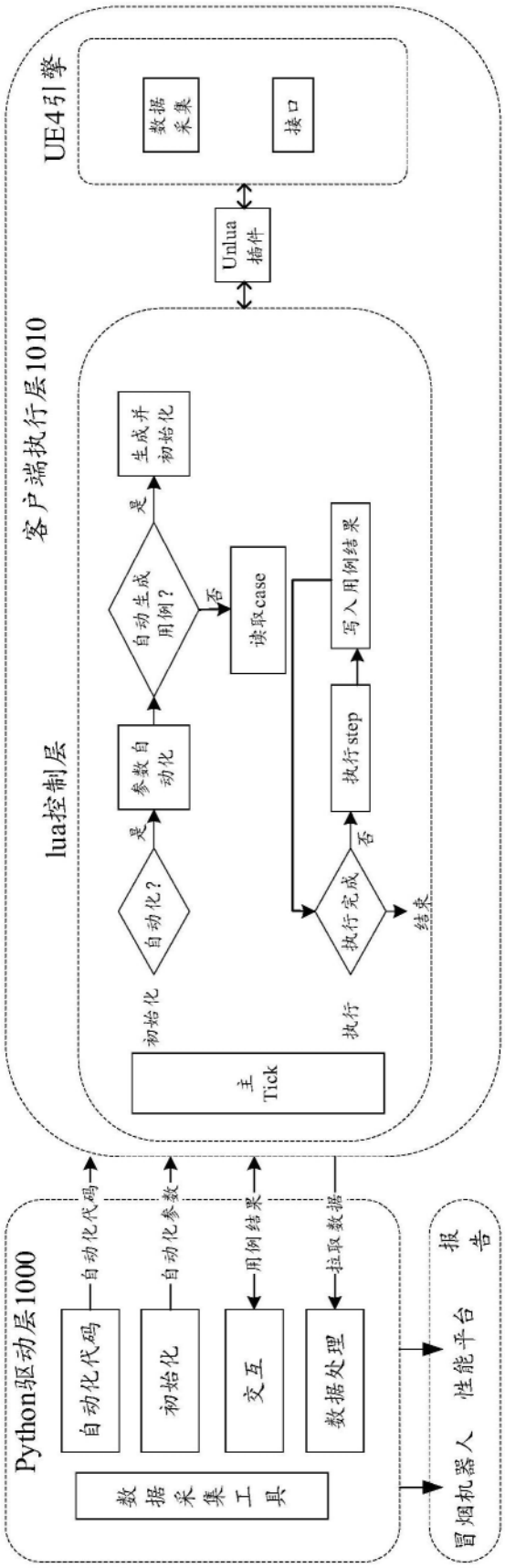

UE4引擎
数据采集
接口
Unlua插件
客户端执行层1010
lua控制层
生成并初始化
是
自动生成用例?
否
读取case
参数自动化
是
自动化?
初始化
主Tick
执行
执行完成
否
结束
执行step
写入用例结果
Python驱动层1000
自动化代码
自动化参数
用例结果
拉取数据
自动化代码
初始化
交互
数据处理
数据采集工具
报告
冒烟机器人 性能平台

图10

```
--lua自动化控制代码示例
zzzzzzzzzz--执行步骤check
zzzzzzzzzz--成功执行了自动测试
zzzzzzzzzz--保存/不保存结果
zzzzzzzzzz
zzzzzzzzzz
```

（a）

```
--自动化接口示例
--下载时间
function aaaaaaaa
    return aaaaaa --返回封装好的自动化测试接口
end
```

（b）

图11

```
--生成用例

function xxxxxxx
    if xx==1 then --商品换装详细遍历
        xxxxxxxx
         xxxxxxxx
    else if xxxx== 2 then --商品换装快速冒烟
        xxxx
        xxxx
    end

    UE4.xxxxxxxx
    return xxxx
end
```

（a）

```
--用例文件示例
yyyyyyy
    yyyyyyy:切换场景
    yyyyyyy:切换分类
    yyyyyyy:执行换装
    yyyyyyy:返回主场景
    yyyyyyy:放弃保存
```

（b）

图12


1300
获取模块1310
功能确定模块1320
接口封装模块1330
用例生成模块1340
用例执行模块1350

图13


1300
获取模块1310
功能确定模块1320
接口封装模块1330
用例生成模块1340
第一生成单元1341
第二生成单元1342
用例执行模块1350
逻辑确定模块1360

图14

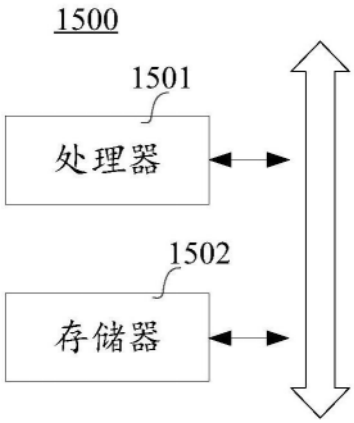


图15