# 作为移动应用程序服务的自动化并行GUI测试

# Amira Ali | Huda Amin Maghawry | Nagwa Badr

Department of Information Systems, Ain Shams University, Cairo, Egypt Correspondence Amira Ali, De-partment of Information Systems, Ain Shams University, Cairo, Egypt.

Email: amira.aly92@live.com

摘要

近年来，由于智能手机的普及和移动应用程序的大量开发，测试移动应用程序受到了越来越多的关注。在发布供公众使用之前，必须测试移动应用程序。图形用户界面（GUI）测试是一种移动应用程序测试，用于确保GUI组件的正确功能。通常，无论是手动的还是自动的，GUI测试都需要大量的工作和时间。云计算是一种新兴的技术，可以应用于软件工程领域，利用云计算资源克服传统测试方法的缺陷。因此，将测试即服务作为一种服务模型引入，以完全自动化的方式执行所有测试活动。本文提出了一种基于测试即服务体系结构的移动应用程序图形用户界面测试系统。系统在多个虚拟节点上执行所有测试活动，包括自动生成测试用例和同时执行测试，以测试基于Android的应用程序。该系统减少了测试时间，满足了移动应用程序的快速上市限制。此外，所提出的系统架构解决了许多问题，例如最大化资源利用率、持续监控以确保系统可靠性、以及应用容错方法来处理任何故障的发生。

关键词

TAAS、云计算、移动应用程序测试、GUI测试、Appium

1 介绍

测试移动应用程序被定义为测试为智能手机开发的软件应用程序的功能和质量的过程。1由于智能手机用户数量的不断增加，移动应用程序已经获得了广泛的欢迎。每天开发数千个移动应用程序，这些应用程序可在许多商店中使用，为用户提供从中选择的大量选项。2但是，由于时间到市场的压力，大多数移动应用程序都是在有限的资源、预算和有限的时间内开发的。这严重影响了移动应用程序开发的质量。3因此，在有限的时间和成本内对移动应用程序进行充分的测试成为了一个巨大的挑战，将影响移动应用程序开发的持续性。4移动应用程序测试需要大量的时间、成本、资源和熟练的测试人员和技 术人员。用于编写和执行测试用例的工具。5-7

在过去几年中，云计算获得了极大的关注。云计算是分布式计算范例的下一次革命，它可以支持按需服务共享，具有更高级别的灵活性和动态可扩展性，可以配置可共享的可共享计算资源8.9因此，将软件测试迁移到云环境并利用云无限存储，资源（硬件和软件）以及可扩展的基础架构是克服传统测试策略难度的最佳解决方案。10-15在基于云的环境中，由于存在容易，因此无需设置测试环境资源 - 需求虚拟测试环境。这有助于满足移动应用程序的快速上市时间限制。因此，测试即服务（TaaS）作为服务模型呈现，自动执行所有测试活动，包括自动测试用例生成，测试执行和测试报告生成。16-18被测应用程序（AUT）被提交到TaaS平台。 TaaS平台自动完成所有测试活动，生成详细的测试报告，并将测试报告提交给用户。

在本文中，提出了一种基于TaaS架构的移动应用程序图形用户界面（GUI）测试系统。所提出的系统具有可扩展且易于使用的架构。解决了导致移动应用程序高效且经济高效的测试的关键因素和挑战。 GUI与移动应用程序交互变得无处不在。因此，移动应用程序的用户关心移动应用程序的GUI控件是否正常运行并按照所需规范工作。19移动应用程序的GUI测试作为一种统一的测试类型，旨在确保AUT满足其功能要求和达到高质量标准。 GUI测试用例由输入事件序列组成，以确保GUI组件符合其所需的规范。基于模型的测试（MBT）被定义为一种测试方法，它从不同的模型中导出测试用例。模型有两种类型：系统需求模型（黑盒测试）和从源代码构建的模型（白盒测试）。20,21

GUI测试用例生成需要对用户行为进行精确建模，以模仿移动应用程序用户与GUI控件之间的交互。本文介绍了基于UML活动图的测试用例生成方法。统一建模语言（UML）是一种广泛使用的建模语言，可以有效地模拟不同的软件工件。22 UML活动图用于根据逐步活动和操作对软件工作流进行建模。活动图成为表示允许自动GUI移动应用程序测试的用户行为的良好候选者.23

在实践中，GUI测试花费太多时间，这可能导致移动应用程序的延迟上市。这就是为什么必须开发自动化的成本效益测试技术作为替代方案。由于Android应用程序的快速增长，24建议的系统侧重于在基于云的环境中对Android应用程序进行GUI测试。此外，Android是一个开源平台，可以为我们的实验提供Android应用程序的可用性。

本文的其余部分安排如下：第2部分重点介绍了在基于云的环境中测试移动应用程序的最相关相关工作。 第3节介绍了建议的系统架构以及其模块功能的详细说明。 第4节介绍了拟议系统的实验结果。 最后，第5节介绍了结论和未来的工作。

2 相关工作

最近，一些研究涉及从不同角度研究基于云的环境中的移动应用测试。一些研究为移动TaaS提供了有关定义，要求，利益，问题和挑战的信息性讨论，如文献所述.25-28

此外，许多研究人员将原型架构引入移动TaaS。例如，Shenbin Zhang等人29提出了一个基于TaaS平台的Android原生应用的移动应用功能测试解决方案。允许用户通过配置设备的上下文参数（如位置，语言和网络速度等）来自定义测试环境。此外，作者还介绍了测试脚本生成的功能遍历方法。功能遍历方法从基于操作特征计算每个操作的权重开始，作为控件的类型，大小，文本和操作的动作。然后，执行具有高权重值的操作。作者专注于遍历活动，其中包括权重超过特定阈值的操作，否则将无法访问活动。这导致生成的测试脚本的不完整覆盖（即，活动覆盖，操作覆盖）。

Riyadh Mahmood等人30提出了一个在云环境中为Android应用程序进行自动安全测试的框架。许多启发式方法用于生成测试用例，以提高在Android应用中发现安全漏洞的可能性。

Tao Zhang等人31提出了一种针对移动应用的兼容性测试服务。介绍了生成兼容性测试序列的移动兼容性测试方法。它基于功能树来建模兼容性功能。特征树中的每个叶节点表示基本兼容性特征。 K-Means算法用于聚类具有类似兼容性功能的移动设备。然后，设备集群根据其市场份额进行排名。此外，作者还讨论了移动兼容性测试所需的基础设施以及移动兼容性测试服务器，该服务器由5层组成：（1）测试租户，（2）测试服务，（3）兼容性测试，（4）沟通，（5）测试数据库。

Tor-MortenGrønli等人32提出了一个名为（Mobilette）的并行跨平台移动测试执行框架的概念原型。但是，Mobilette只能测试基本的移动应用程序，因为它支持很少的基本GUI操作。此外，Mobilette框架仅执行测试执行而不涉及其余的测试活动，尤其是测试用例生成，这是测试过程的核心阶段。

Oleksii Starov等33提供了移动系统云测试（CTOMS）的综合视图。 CTOMS是一个用于移动开发的分布式TaaS平台。所提出的系统具有以下层：表示层，平台层和云解决方案的信息逻辑层。主应用程序用于表示平台层，例如：Google App Engine云和可选的Hadoop实例层，它利用MapReduce算法在节点之间分配测试并收集结果。主应用程序向最终用户和从属节点计算机提供演示。信息层由云数据存储器表示。作者向CTOMS介绍了整体架构。但是，没有讨论用于执行测试过程中涉及的每个测试活动的技术。

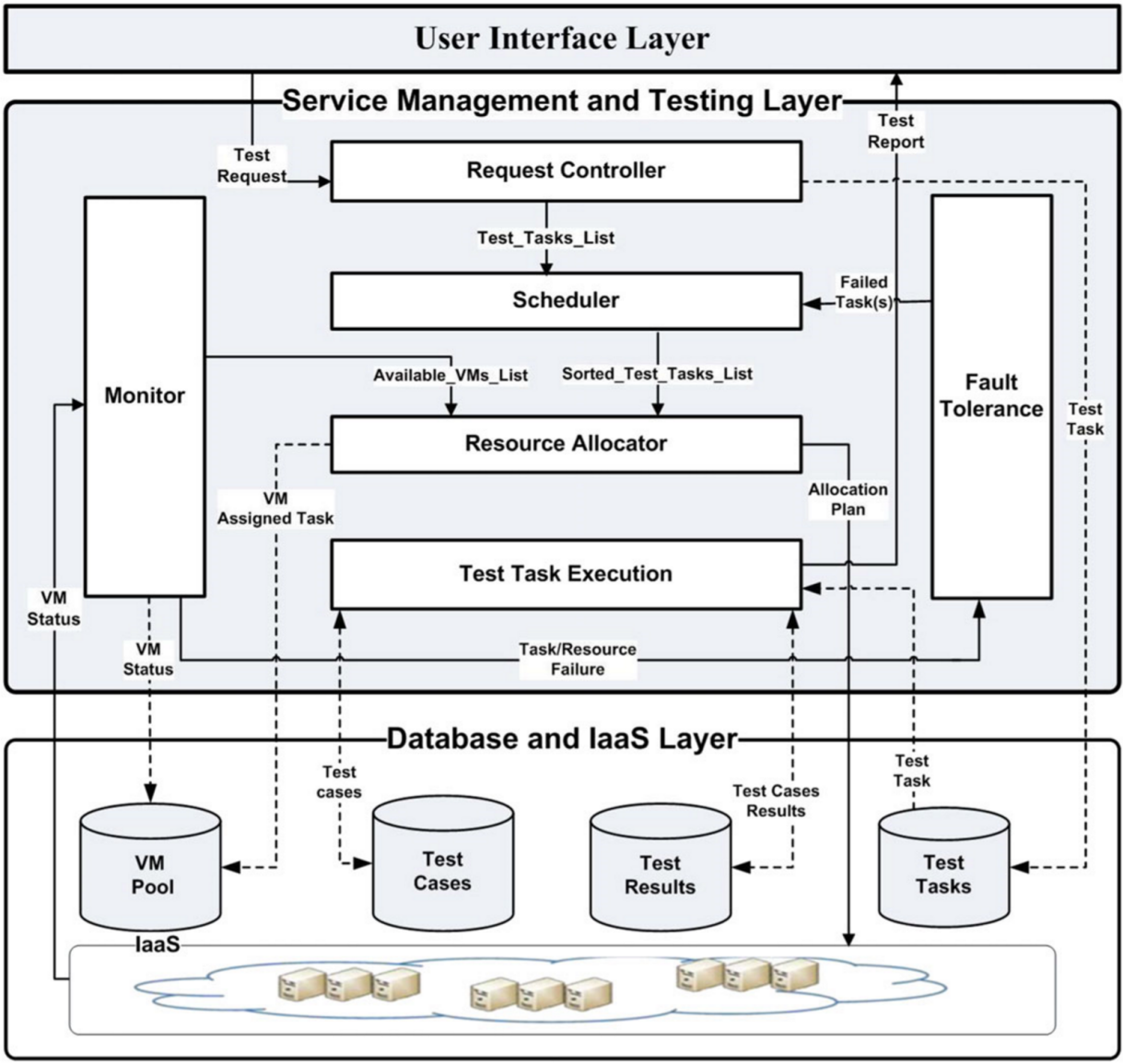
有许多基于商业云的移动测试工具和服务，如：Xamarin Test Cloud和Perfecto Test。 Xamarin Test Cloud34为跨平台移动应用程序测试提供了许多测试工具，并提供了数千个设备来运行测试用例。 XamarinTest Cloud对开源库的访问权限有限。 PerfectoTest35允许在不同平台（例如，Android，iOS）上对移动应用程序进行实时测试。尽管Xamarin Test Cloud和Perfecto Test在测试执行方面具有优势，但两种工具都不会执行自动测试用例生成。

**有几项研究涉及基于UML图的自动测试用例生成，用于测试移动应用程序。 Ang Li等人36提出了一个名为ADAutomation的移动应用程序测试框架。 ADAutomation框架支持用户行为建模，测试用例生成，测试执行以及测试后分析和调试。作者介绍了基于UML活动图的测试用例生成方法。 Anbunathan R等人37介绍了一种在解析序列图之后生成测试用例的方法，然后生成基于XML的测试用例以及随后基于APK的Android手机测试脚本。虚拟测试工程师是基于此方法开发的工具。虚拟测试工程师用于测试多个Android应用程序。但是，之前的2个框架由于在1台机器上执行顺序测试而具有较长的总体完成时间（即，包括测试用例生成和测试执行）。**

因此，需要一种基于TaaS架构的集成可扩展系统。可以最大限度地利用云资源的系统可以有效地自动化测试过程并保证高测试覆盖率。

3 提议的架构

在本文中，提出了一种用于GUI移动应用程序TaaS的体系结构。 用户可以向拟议的TaaS提交请求，以便自动有效地执行整个测试过程。 图1显示了所提出的GUI移动应用程序TaaS的整体架构。 所提出的架构的3层如下：（1）用户界面层，（2）服务管理和测试层，以及（3）数据库和基础设施即服务（IaaS）层。



**图1** 移动应用程序GUI TaaS体系结构

A.用户接口层：

它是建议架构中的第一层。用户界面层是一个Web前端，允许用户通过互联网与建议的TaaS架构进行交互。用户提交测试请求并定义请求要求（如请求截止日期（即完成AUT测试的最终时间））。最后，用户收到测试结果报告。提交给建议架构的用户输入的详细描述将在下一节中介绍。

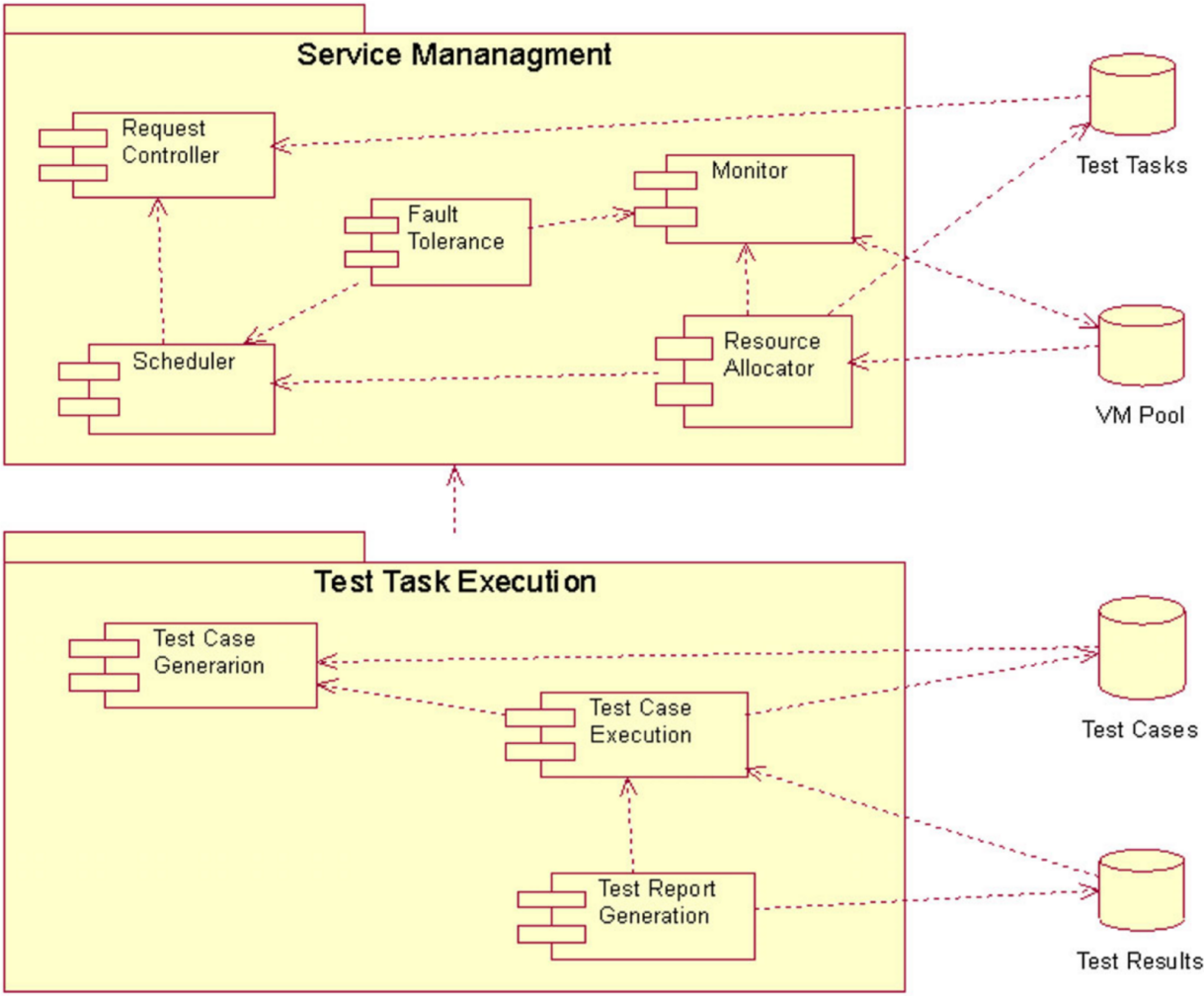
B.服务管理和测试层：

该层包括所提出的架构的主要功能。服务管理和测试层的主要模块是：请求控制器，调度器，监视器，资源分配器，容错和测试任务执行模块。每个模块的详细描述将在下一节中介绍。

C.数据库和IaaS层：

该层包括4个存储库：（1）测试任务存储库存储测试请求及其截止日期和完成测试过程所需的输入数据，例如AUT的活动图，稍后将详细描述。 （2）测试用例存储库存储生成的测试用例。 （3）测试结果存储库存储执行的测试用例的结果。 （4）虚拟机（VM）池存储关于每个VM的信息，例如VM状态，VM负载和分配的任务。通过虚拟化技术向用户提供所有必需的资源。因此，存在IaaS层以包括执行测试任务的VM。

图2显示了所提出的体系结构的组件与将这些组件组织到包中之间的依赖关系。组件和包之间的虚线表示依赖关系。依赖关系意味着1个组件的功能取决于另一个组件提供的功能。朝向数据存储库的虚线意味着将组件输出存储在数据存储库中。而来自数据存储库的虚线意味着检索完成组件功能所需的数据。建议的系统体系结构中的组件分为两个主要包：服务管理包和测试



**图2**建议的体系结构组件图

任务执行包。两个包之间存在依赖关系，因为测试任务执行包依赖于服务管理包中包含的组件提供的功能。 测试任务执行包包括以下组件：测试用例生成，测试用例执行和测试报告生成。 这些组件负责执行整个移动应用程序GUI测试过程。 服务管理包包括以下组件：调度程序，监视器，资源分配器，容错和请求控制器。 服务管理包的组件负责管理所提议的系统体系结构中包含的测试任务和资源。 所提出的系统中每个组件的功能和工作方法将在下一节中介绍。

3.1 服务管理和测试层

本节为建议的体系结构中的服务管理和测试层中的每个模块提供了深刻的描述。每个模块的功能讨论如下。

A.Request控制器模块

它是服务管理和测试层中的第一个模块。用户通过用户界面向建议的系统提交测试请求。请求控制器将测试请求转换为测试任务，并将任务存储到Test Tasks存储库中。然后，请求控制器模块将任务添加到Test\_Tasks\_List。

B.Scheduler模块

调度程序模块定义测试任务的执行顺序。在提出的体系结构中，调度程序模块实现为本地服务，该服务应用排队方法来确定Test\_Tasks\_List中每个任务的优先级。调度程序从请求控制器接收Test\_Tasks\_List。排队方法考虑任务截止时间和等待时间来计算任务优先级.38任务截止时间是用户完成AUT测试的最终时间，而任务的等待时间是当前时间和任务时间之间的持续时间是由用户提交的。然后，调度程序模块根据其优先级按降序排列任务。任务根据任务截止日期按升序排序（即，具有较早期限的任务将具有更高的优先级）。然后，具有相同期限的任务将根据其等待时间进行排序（即，具有较高等待时间的任务将具有较高优先级）。

C.监控模块

为了实现更高级别的系统可靠性，监视模块作为本地服务用于跟踪所有任务和资源的实时状态。它将这些信息存储在VM池中，如表所示。 1.监视器模块通过检索VM池中存储的每个VM的状态来确定Available\_VMs\_List。然后，监视器模块将Available\_VMs\_List转发到资源分配器模块。此外，监控模块检测故障，无论是任务故障还是资源故障，然后将故障详细信息转发到容错模块进行管理。

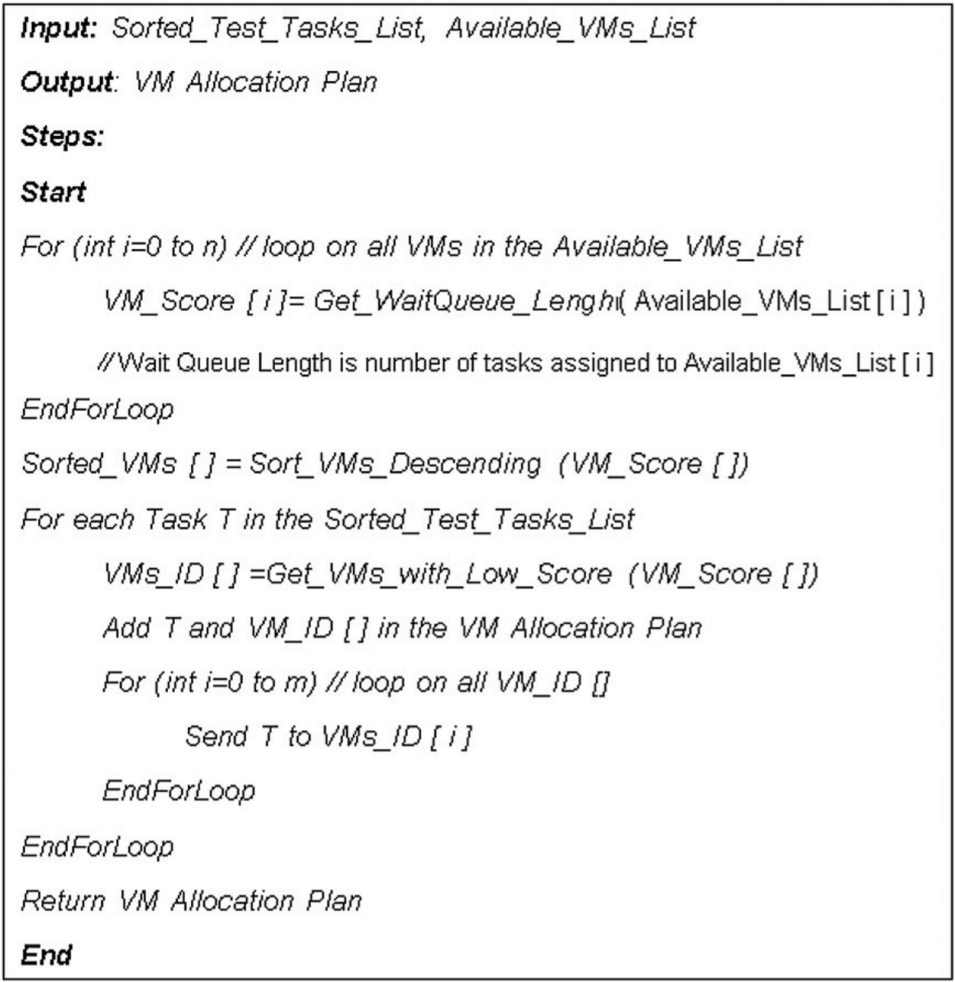
D.资源分配器模块

在基于云的环境中，有必要最大化资源利用率并改善系统负载平衡.39因此，存在资源分配器模块。资源分配器模块将任务分配给可用虚拟机以执行任务，从而提高硬件和软件资源利用率并保证负载平衡。执行任务后，将释放其相关资源以执行其他任务。资源分配器模块分别从调度程序和监视器模块接收Sorted\_Test\_Tasks\_List和Available\_VMs\_List。然后，资源分配器模块根据图3中引入的动态生成的分配计划将任务分配给VM。

如图3所示，每个VM的得分根据其等待队列的长度计算。 然后，根据计算的分数以降序排列VM。 具有高优先级的任务被分配给具有低分数的VM。 在基于云的环境中，测试任务可以分解为在多个VM上执行的多个子任务，以缩短整体测试时间。40因此，为测试每个AUT而生成的测试用例在分配的VM上同时执行。

TABLE 1 VM 池

|  |  |  |
| --- | --- | --- |
| VM ID | VM State | VM Load |
| (number of assigned tasks) |
| 1 | ON | 3 |
| 2 | ON | 5 |
| 3 | OFF | 0 |



**图3**分配计划生成方法

E.容错模块

此模块用于处理在任务执行期间发生的故障。如前所述，监视器模块跟踪VM的实时状态和任务执行状态以检测故障。监视器模块捕获的故障详细信息将转发到容错模块。容错模块根据以下内容处理故障：

1.如果在任务执行期间发生故障。然后，将失败的任务返回到调度程序模块以进行重新调度。

2.如果在VM中发生故障。然后，分配给故障VM的所有任务将返回到调度程序模块以进行重新调度。

F.测试任务执行模块

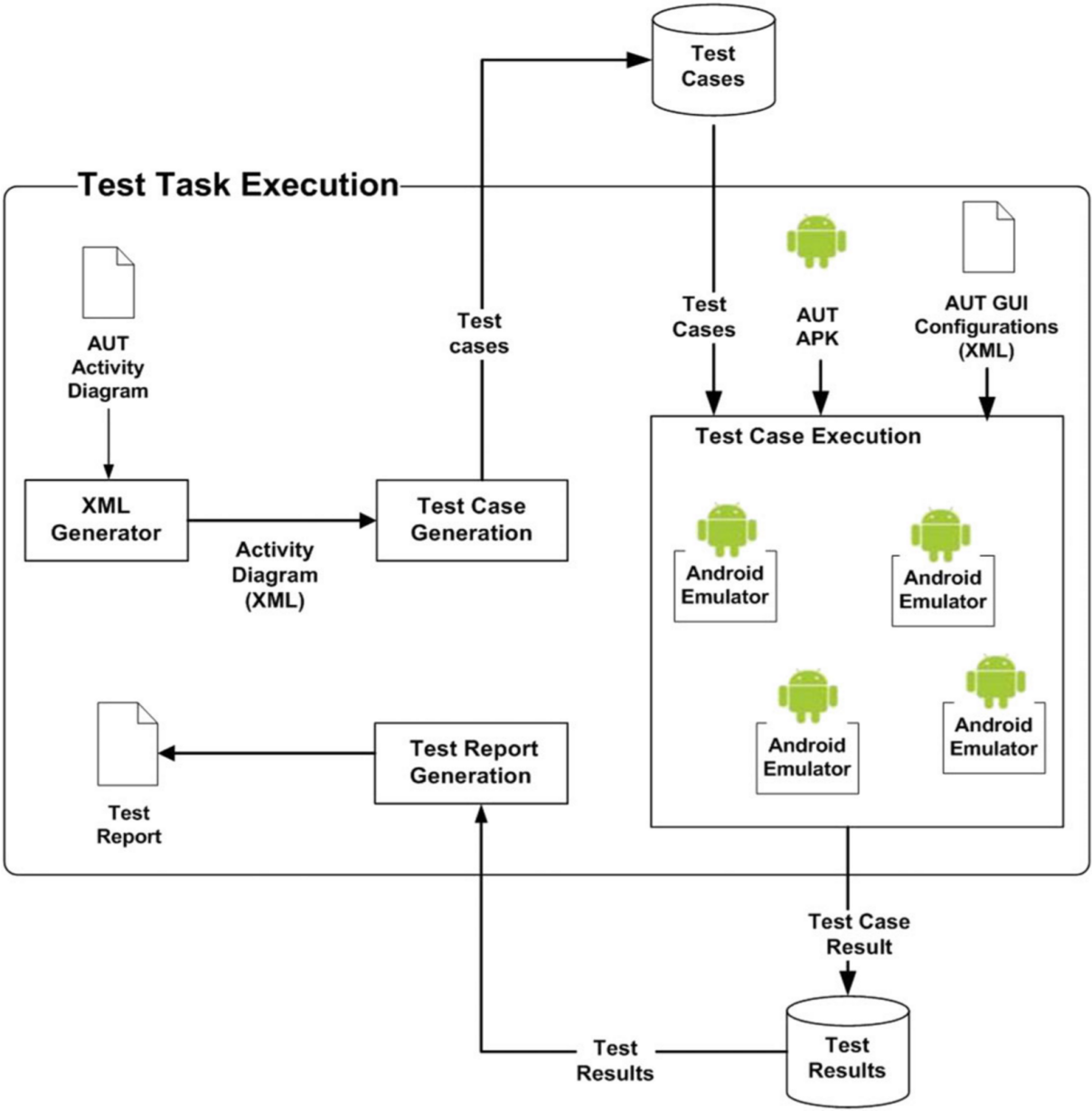
该模块负责在基于云的环境中为Android应用程序自动执行整个GUI测试过程。它旨在通过利用云资源来减少整体测试时间和工作量。该模块采用有效的测试方法来提高测试质量和可靠性。

图4显示了Test Task Execution模块。所提出的系统接收通过用户界面层自动执行整个测试过程所需的输入数据。测试任务执行模块的输入是UML活动图，Android应用程序包（APK）和AUT的GUI配置文件。 UML图广泛用于建模应用程序需求。因此，UML图被认为是设计测试用例的重要来源之一.22活动图可以表示各种控制流和GUI动作之间的依赖关系。这就是为什么活动图成为建模用户行为以允许移动应用程序的自动GUI测试的良好候选者.36 APK是一个Java字节码包，用于分发和安装Android应用程序。它允许测试移动应用程序而无需应用程序源代码。 AUT配置文件包括GUI布局信息：GUI小部件id，触发GUI小部件和小部件类型所需的事件（例如，编辑文本或按钮）。

AUT的GUI配置文件用于允许建议的系统捕获自动测试用例生成和执行所需的GUI交互。

测试任务执行模块中的流程如下：

·首先，XML Generator将在UML绘图工具（例如，Visual Paradigm41）上绘制的活动图转换为基于XML的活动图文件。 获得的基于XML的活动图文件是建立自动化测试方法的基 本单元。 使用XML解析器工具（例如，Visual Paradigm41）将AUT活动图转换为基于XML的活动图文件。



**图4** 测试任务执行模块

·测试用例生成模块根据从获取的XML文件中提取的信息自动生成一组GUI测试用例。生成的测试用例存储在Test Cases存储库中。每个测试用例包括在AUT的GUI小部件上执行的一系列动作。

·然后，在基于云的环境中，在运行Android Emulator的多个虚拟节点上同时执行测试用例。测试用例执行模块在执行测试用例之前在Android模拟器上安装AUT的APK，然后执行测试用例并将测试用例的结果存储在测试结果存储库中。

·最后，测试报告生成模块生成测试报告，包括执行的测试用例的结果并将其提交给用户。

测试任务执行模块的每个子模块将在下面更详细地解释：

1. 测试用例生成模块

测试用例生成是任何测试过程的核心阶段。使用AUT时，测试用例应模仿完整的用户行为。在GUI测试中，每个测试用例包含在AUT的GUI小部件上执行的一系列相关操作。因此，确定GUI动作之间的依赖关系是GUI测试用例生成模块中的关键过程。移动应用程序可以使用不同类型的模型进行建模。21,37,42-44 MBT作为一种可行的方法，以有效和高效的方式测试移动应用程序。45因此，MBT技术可以实现Android应用程序的GUI测试，需要在详细程度描述AUT的正式模型。

图5显示了一种动态自动化方法，用于在将其解析为基于XML的文件后使用AUT的活动图生成测试用例。首先，生成依赖表以包括从基于XML的文件中提取的所有信息。依赖关系表中的每一行代表活动图中的元素。生成的依赖关系表有4列：

1.First列包含元素名称。

2.第二列包括元素类型（即，动作，决策，初始和最终节点）。

3.第三列包含该元素的所有边的列表。

4.第四列包括该元素的边缘。

边缘用于确定元素之间的依赖关系。元素之间的依赖性由1个元素的边缘输出的定义来定义，该元素与当前元素的边缘相同。

然后，基于依赖性表生成有限状态机（FSM）46图。每个节点表示依赖关系表中的条目。根据依赖关系表中定义的元素之间的依赖关系确定从1个节点到另一个节点的转换。图的初始和终端节点分别是活动图的初始和最终节点。然后，使用深度优先搜索算法47遍历生成的图以找到所有可能的独立路径。独立路径21被定义为从起始节点到终端节点的任何路径，其引入至少1个之前未被遍历的新边缘。每个路径由相关GUI动作序列组成，表示使用AUT时的用户行为。

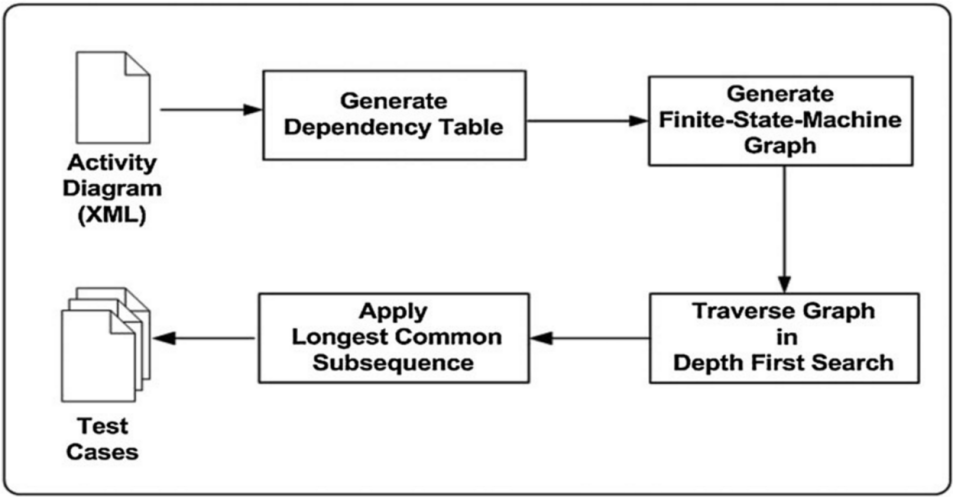
最后，最长公共子序列算法（LCS）48应用于导出路径。 LCS找到总是一起出现的最长的GUI动作的常见序列。在派生的基础路径上应用LCS的原因是删除作为另一个路径中的子路径包含的测试路径。这减少了测试路径的数量并保证了高覆盖率。因此，每个GUI组件至少运用一次。应用LCS后获得的路径代表测试用例。

ii. 测试用例执行模块

生成的测试用例以及AUT配置文件用于获取可执行的测试脚本。使用Appium执行测试脚本.49 Appium是一个移动自动机测试工具，具有许多优点：它是一个开源工具，它不需要修改AUT，因此无需重新编译AUT，并且它通过运行测试支持可扩展性多个模拟器.50,51还有许多其他移动自动机测试工具，如：Robotium和Calabash。 Robotium52一次只能在1个设备上执行测试执行。 Calabash53不支持复杂的测试场景。这就是为什么与其他移动测试工具相比，Appium被认为是所提出系统的最佳选择。在提议的系统中，同时测试用例执行在基于云的环境中的多个虚拟节点上进行，其中Appium安装在运行Android Emulator的每个虚拟节点上。在执行测试用例之前，将AUT的APK提交到指定的虚拟节点并安装在他们的Android仿真器上。

iii. 测试报告生成模块

最后，所提出的系统生成测试报告并通过用户界面将其提交给用户。测试报告包括从多个虚拟节点收集的测试用例结果。测试报告自动组织测试结果信息。测试报告包括自动生成的测试用例的步骤以及每个测试用例的执行结果，无论是失败还是通过。这对于用户确定失败的测试以解决AUT问题很有价值。



**图5** 自动测试用例生成方法

Test Task Execution模块的活动图如图6所示.Test Task Execution模块的总体流程如下：

1. Test Task Execution模块从Test Tasks存储库中检索AUT的活动图。因为Test Tasks存储库存储用户提交的测试请求以及完成测试过程所需的输入数据。输入数据包括第3.1节中提到的AUT的活动图，APK和GUI配置文件。
2. 使用XML解析器工具（例如，Visual Paradigm）将AUT的活动图转换为基于XML的活动图文件。
3. Test Case Generation Module生成依赖表以存储从基于XML的活动图文件中提取的所有信息。
4. 如第3.1节所述，基于依赖关系表中包含的信息生成FSM图。
5. 深入遍历FSM图，首先搜索所有可能的测试路径。
6. 最长公共子序列算法（LCS）应用于提取的测试路径，以去除作为另一路径中的子路径包括的路径。应用LCS后获得的路径代表测试用例。
7. 测试任务执行模块从测试任务存储库中检索AUT的APK文件和GUI配置文件。需要APK文件和GUI配置文件来自动执行生成的测试用例。
8. 测试任务执行模块将AUT的APK文件提交给指定的虚拟节点，其中将使用Appium工具执行测试用例。
9. Test Task Execution模块在分配的虚拟节点的Android Emulators上安装APK文件。
10. Test Task Execution模块使用Appium工具在多个虚拟节点上同时执行生成的测试用例。 AUT的GUI配置文件包括GUI布局信息，如：GUI小部件id，触发GUI小部件和小部件类型所需的事件（例如，编辑文本或按钮）。 AUT的GUI配置文件允许访问AUT的用户界面元素（UIElements）。
11. Test Task Execution模块将测试结果存储在Test Results存储库中。
12. Test Task Execution模块在多个虚拟节点上同时执行测试用例。
13. 当测试任务执行模块完成所有测试用例的执行时，将生成测试报告以包括执行的测试用例的结果。测试报告通过用户界面提交给用户。

4 试验结果

进行实验以评估所提出的系统的可行性。四个虚拟机用于模拟基于云的环境，并允许同时处理测试请求。实验中使用的4个虚拟机是在1台物理机上创建的。 VMware54是一种允许用户通过使用虚拟化技术创建云环境的软件。 VMware workstation55是行业标准，允许用户在1台物理计算机上创建多个VM。MS SQL Server56用于创建以下4个存储库：测试任务存储库，测试用例存储库，测试结果和VM池。 Appium安装在VM上。在开始测试用例执行之前，将在Android模拟器上安装AUT APK文件。

A.实验1：

本实验的目的是根据行动覆盖范围和活动覆盖率评估拟议的测试用例生成方法。操作范围定义为测试操作与活动图中所有操作之间的比率。活动范围定义为测试活动与活动图中所有活动之间的比率。将所提出的方法与功能遍历方法29和MonkeyRunner进行比较。功能遍历方法是在TaaS平台中使用的移动应用程序测试方法。功能遍历为每个操作分配权重。然后，它遍历重量超过特定阈值的操作。 MonkeyRunner基于随机事件来测试AUT。在本实验中，选择了2个应用程序作为被测应用程序：简单记事本应用程序58和时钟应用程序

简单的记事本应用程序是一个原生的Android应用程它允许用户轻松记笔记。它具有许多功能，如创建新的清单，排序，搜索等。简单的记事本应用程序包含6个GUI视图。视图1表示主视图，用户可以选择添加新笔记，添加新检查表，调整应用程序设置，添加新文件夹，搜索或排序已保存的笔记。 View 2允许用户通过编写标题和正文来创建新笔记。视图3显示了新的清单视图。 View 4允许用户添加新文件夹。视图5显示搜索或排序结果。最后，视图6允许用户调整应用程序设置。附录部分中的图A.1显示了使用UML活动图的Simple Notepad应用程序的工作流程。活动图用于为Simple Notepad应用程序的面向GUI的用户行为建模，以便自动生成测试用例。每个GUI视图都被建模为所示活动图中的活动。活动图中的每个动作对应于在GUI控件上执行的事件。

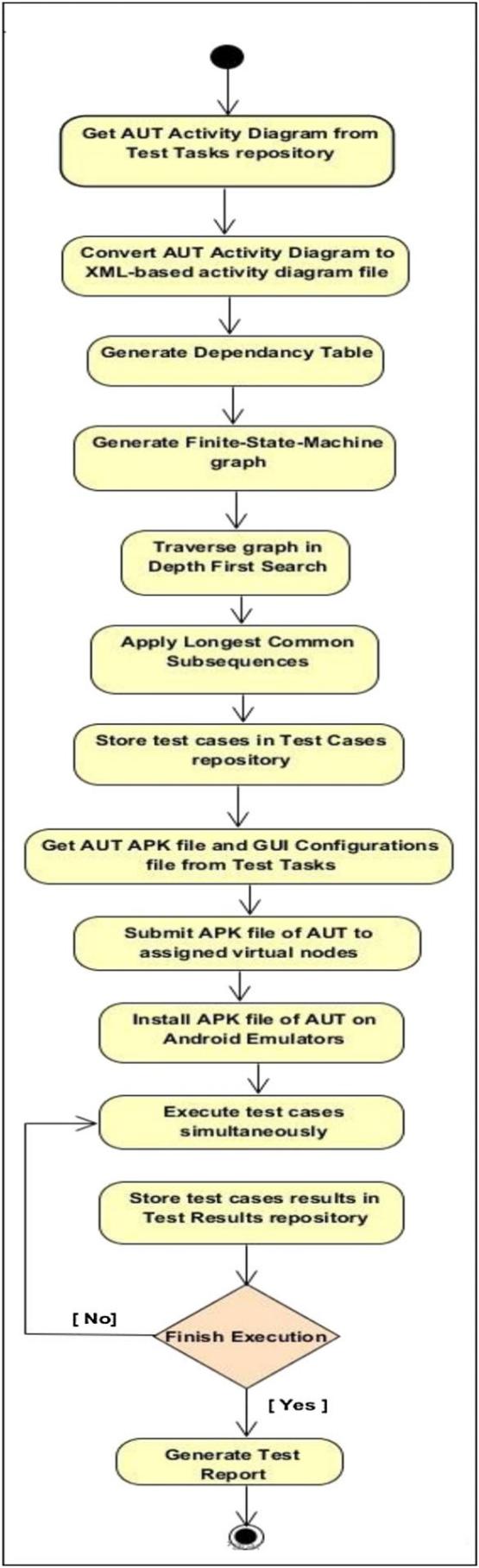


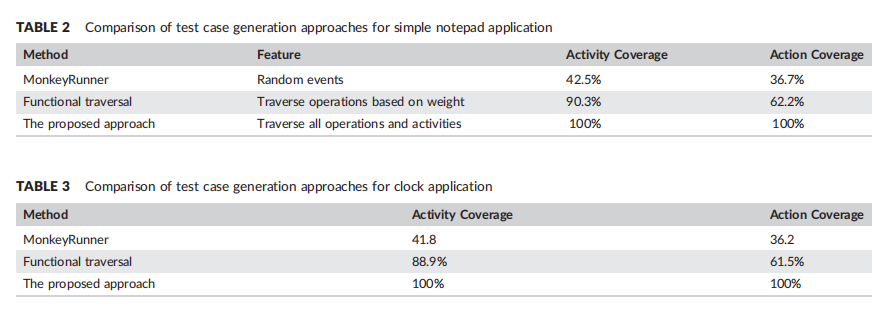
图6 测试任务执行模块活动图

活动图，AUT配置文件和Simple Notepad.apk文件都通过用户界面层提交到TaaS平台。首先，根据先前解释的测试用例生成方法自动生成测试用例。生成的测试用例数为85个测试用例;每个测试用例表示GUI用户遍历行为。生成的测试用例涵盖所有AUT活动（即GUI视图）和操作。建议的测试用例生成方法在菜单小部件中的每个项目上循环。因此，每个GUI控件和AUT中任何菜单小部件中的每个项目至少运行一次。如表2所示，与其他两种方法相比，所提出的方法具有更高的活动性和行动覆盖率。

类似地，Clock应用程序用作AUT来验证建议的测试用例生成方法。时钟应用程序包括许多功能，如设置警报，添加计时器，运行秒表，以及使用世界时钟跟踪世界各地的时间。附录部分中的图A.2显示了使用UML活动图的Clock应用程序的工作流程。时钟应用程序包含7个GUI视图。视图1表示主视图，用户可以选择添加新警报，添加世界时钟，启动秒表，添加计时器，删除警报或显示时钟应用程序信息。 View 2允许用户通过设置闹钟小时/分钟/秒来创建新闹钟。视图3允许用户选择城市然后添加所选城市的时钟。 View 4允许用户启动或停止秒表。 View 5允许用户通过定义计时器小时/分钟/秒来添加新计时器。 View 6允许用户删除某些警报。最后，视图7显示有关时钟应用程序的信息。每个GUI视图对应于活动图中的活动。建议的测试用例生成方法自动生成9个测试用例。如表3所示，与其他方法相比，所提出的方法保证了更高的活动覆盖率和行动覆盖率。

B.实验2：

此实验的目的是评估在将许多请求发送到TaaS平台与顺序执行时，利用云环境资源对同时测试用例执行的影响。评估是根据执行所有提交的请求所需的总体完成时间来完成的。将所提出的系统与ADAutomation框架进行比较.36 ADAutomation框架基于UML活动图生成测试用例。它在一台机器上顺序执行测试用例。虽然建议的系统在多个VM上同时执行测试请求。图7显示，与ADAutomation框架相比，所提出的系统具有更低的总体完成时间，特别是在测试请求数量增



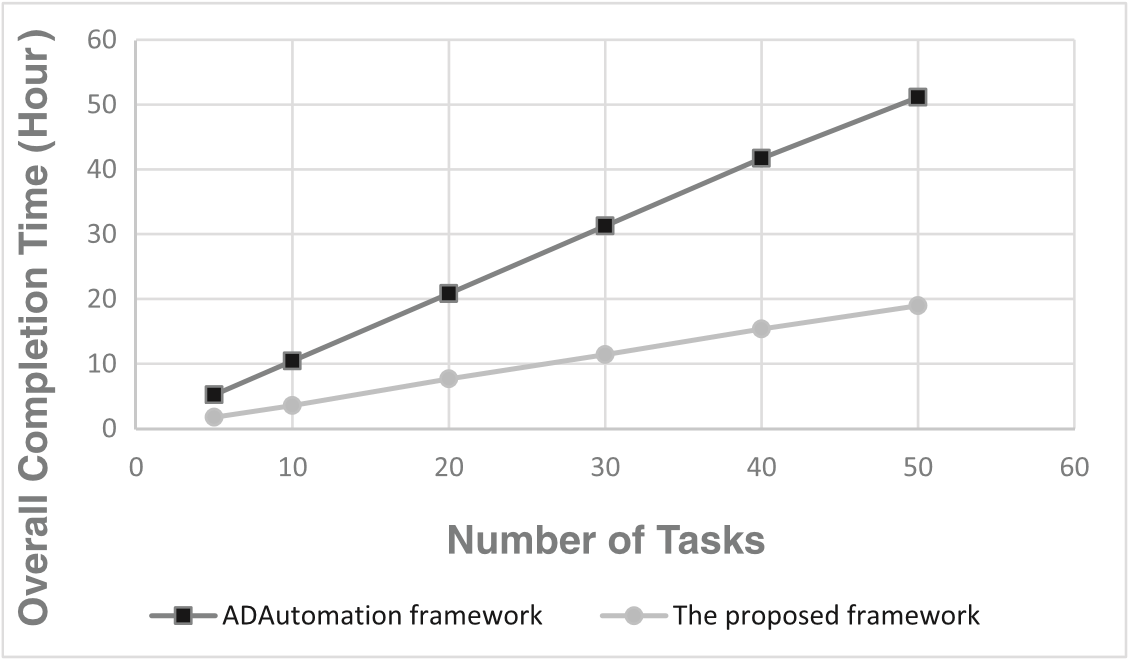


FIGURE 7 Overall completion time comparison for several test request

加时，测试请求从一开始就进入两个系统。这就是为什么两条线看起来都是线性的并且从原点开始。由于以下原因，所提出的系统具有较低的完成时间：（1）基于云的环境提供的可扩展性允许在多个VM上同时执行测试，（2）由调度器模块应用的任务优先级方法，其保证满足测试请求截止日期和减少任务等待时间，（3）由资源分配器模块执行的动态生成的分配计划，其最大化资源利用率并保证系统负载平衡，以及（4）在所提出的系统中使用的测试用例生成方法，其减少了数量通过应用LCS算法自动生成测试用例，以及保证AUT的高测试覆盖率。

5 结论和今后的工作

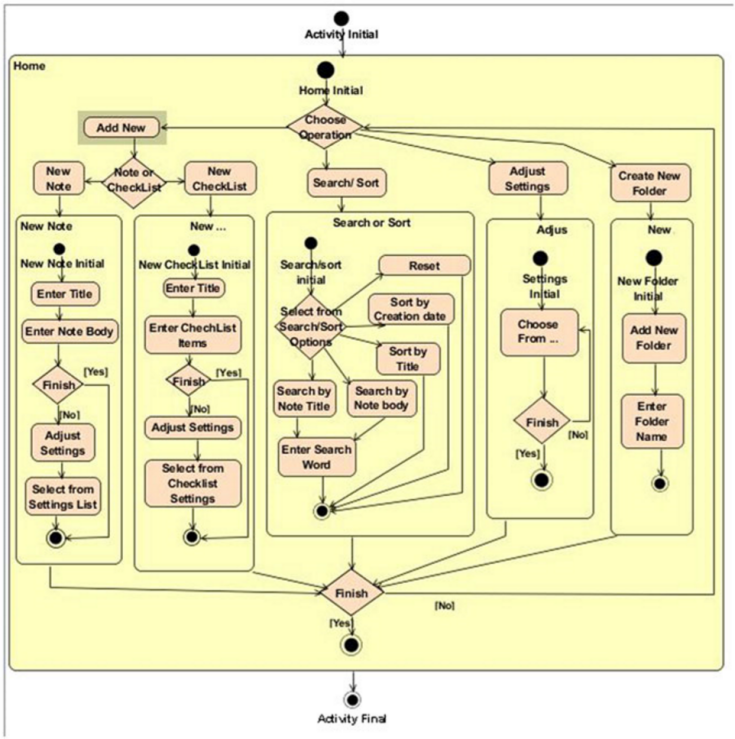
TaaS提供了通过利用云资源和基础架构提供测试即服务的机会。这样可以显着降低测试所需的成本，时间和精力。由于移动应用程序的普及，测试移动应用程序的必要性增加。在发布公共应用程序之前，GUI测试得到了很多关注，以确保移动应用程序的可靠性和功能。因此，本文介绍了一种基于TaaS架构的移动应用GUI测试系统，其中所有测试程序都以完全自动化的方式执行。与其他方法相比，所提出的系统保证了自动生成具有更高覆盖范围的测试用例。此外，通过调度程序和资源分配模块实现基于云的资源负载平衡。资源分配器模块执行动态生成的分配计划，以最大化资源利用率。调度程序模块对任务进行优先级排序，以确保满足测试请求的最后期限并减少任务等待时间。实验结果表明，由于在多个虚拟节点上同时执行测试用例，所提出的系统获得了较高的测试执行效率。此外，与其他方法相比，所提出的系统保证了高测试覆盖率。在未来的工作中，建议的系统可以扩展到包括许多方向，如（1）包括更多类型的移动应用程序测试作为安全测试和回归测试，（2）允许对在不同平台上运行的不同类型的移动应用程序进行GUI测试，（3）使用像Windows Azure这样的真实云环境，通过租用更多数量的虚拟机，为所提议的系统提供更大的可扩展性。

参考文献

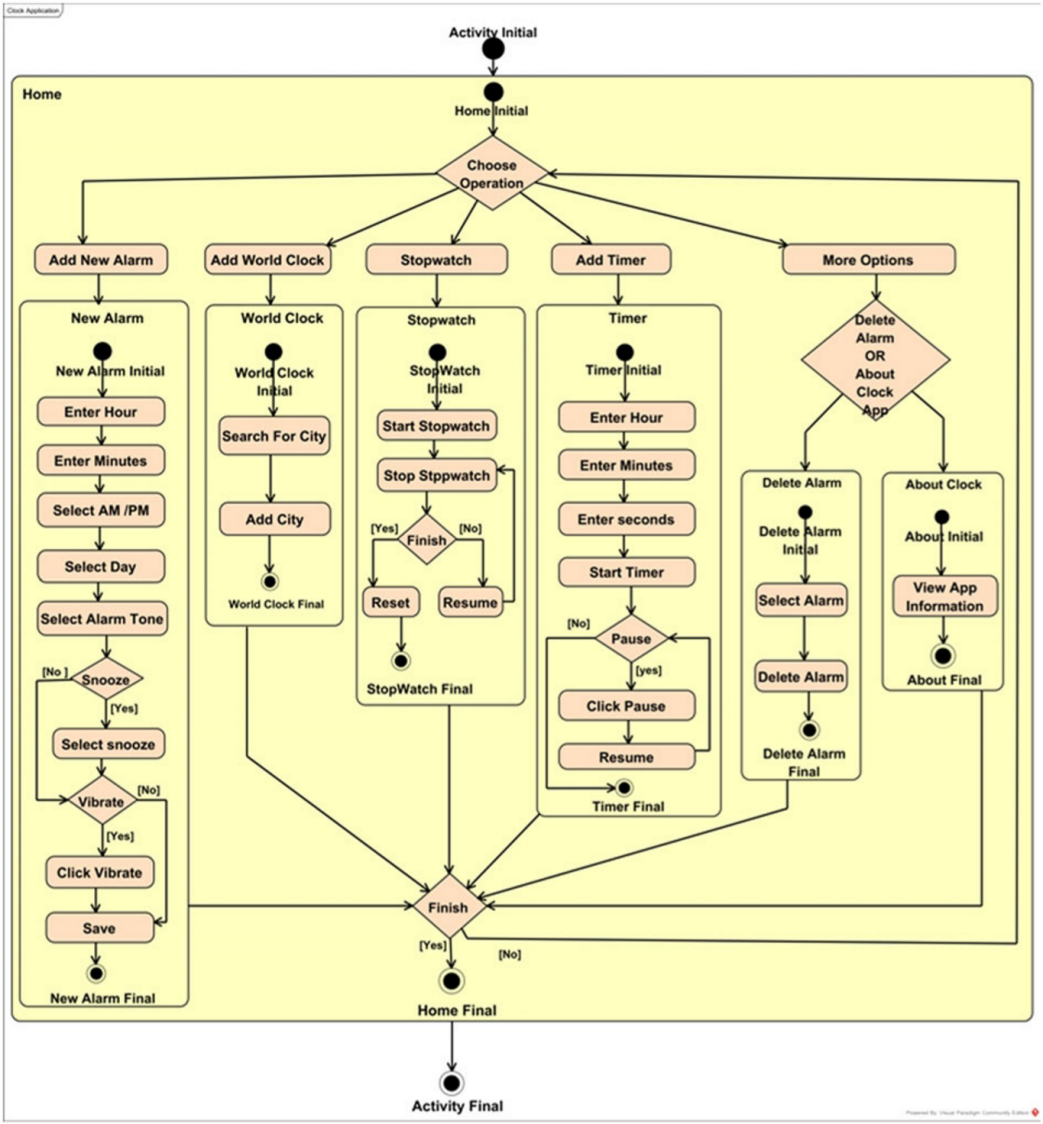
1. Gao J, Bai X, Tsai W‐T, Uehara T. Mobile application testing: a tutorial. Computer. 2014;47(2):46‐55.
2. <https://www.alliedmarketresearch.com/mobile-application-market>
3. Kochhar PS, Thung F, Nagappan N, Zimmermann T, Lo D. Understanding the test automation culture of app developers. in software testing, verification and validation (ICST), 2015 IEEE 8th international conference on, pp. 1‐10. IEEE, 2015.
4. Flora HK, Wang X, Chande SV. An investigation into mobile application development processes: Challenges and best practices. Int J Mod Educ Comput. Sci. 2014;6(6):1.
5. Nagappan M, Shihab E. Future trends in software engineering research for mobile apps. in software analysis, evolution, and reengineering (SANER), 2016 IEEE 23rd international conference on, vol. 5, pp. 21‐32. IEEE, 2016.
6. Kirubakaran B, Karthikeyani V. Mobile application testing—challenges and solution approach through automation. in Pattern Recognition, Informatics and Mobile Engineering (PRIME), 2013 International Conference on, pp. 79‐84. IEEE, 2013.
7. Linares‐Vásquez M, Moran K, Poshyvanyk D. Continuous, evolutionary and large‐scale: a new perspective for automated mobile app testing. in Software Maintenance and Evolution (ICSME), 2017 IEEE International Conference on, pp. 399‐410. IEEE, 2017.
8. Mell P, Grance T. The NIST definition of cloud Comput Secur (2011).
9. Buyya R, Yeo CS, Venugopal S, Broberg J, Brandic I. Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility. Future Generation Computer Systems. 2009;25(6):599‐616.
10. Parveen T, Tilley S. When to migrate software testing to the cloud?. in software testing, verification, and validation workshops (ICSTW), 2010 third international conference on, pp. 424‐427. IEEE, 2010.
11. Tilley S, Parveen T. SMART‐T: migrating testing to the cloud. In Software Testing in the Cloud, pp. 19‐35. Springer Berlin Heidelberg, 2012.
12. Zhenlong P, Zhonghui OY, Youlan H. The application and development of software testing in cloud computing environment. In Computer Science & Service System (CSSS), 2012 International Conference on, pp. 450‐454. IEEE, 2012.
13. Priyadharshini V, Malathi A. Survey on software testing techniques in cloud computing. CoRR, abs/1402.1925, 2014.
14. Riungu LM, Taipale O, Smolander K. Research issues for software testing in the cloud. in Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on, pp. 557‐564. IEEE, 2010.
15. Katherine AV, Alagarsamy K. Software testing in cloud platform: a survey. International Journal of Computer Applications. 2012;46(6):21‐25.
16. Harikrishna P, Amuthan A. A survey of testing as a service in cloud computing. in computer communication and informatics (ICCCI), 2016 international conference on, pp. 1‐5. IEEE, 2016.
17. Gao J, Bai X, Tsai W‐T, Uehara T. Testing as a service (TaaS) on clouds. in service oriented system engineering (SOSE), 2013 IEEE 7th international symposium on, pp. 212‐223. IEEE, 2013.
18. Wang W, Zhang X, Chen T, Wu X, Li X. Cloud computing based software testing framework design and implementation. Advances in Computer, Communication, Control and Automation 2012: 785‐790.
19. Amalfitano D, Fasolino AR, Tramontana P. A gui crawling‐based technique for android mobile application testing. in Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, pp. 252‐261. IEEE, 2011.
20. Berenbach B, Paulish D, Kazmeier J, Rudorfer A. Software and systems requirements engineering: in practice. McGraw‐Hill, Inc., 2009.
21. Wasnik C, Lingam C. Software Testing and Software Development Lifecycles. Int J Comput Distrib Sys. 2013;2(3).
22. Rumbaugh J, Jacobson I, Booch G. The Unified Modelling Language User Guide. Reading MA: Addison‐Weslley; 1999.
23. Chen M, Mishra P, Kalita D. Coverage‐driven automatic test generation for UML activity diagrams. In Proceedings of the 18th ACM Great Lakes symposium on VLSI, pp. 139‐142. ACM, 2008.
24. Surpassed, Gartner Says Annual Smartphone Sales. Sales of feature phones for the first time in 2013. 2014.
25. Murugesan L, Balasubramanian P. Cloud based mobile application testing. in computer and information science (ICIS), 2014 IEEE/ACIS 13 th international conference on, pp. 287‐289. IEEE, 2014.
26. Gao J, Tsai W‐T, Paul R, Bai X, Uehara T. Mobile testing‐as‐a‐service (MTaaS)—infrastructures, issues, solutions and needs. in high assurance systems engineering (HASE), 2014 IEEE 15th international symposium on, pp. 158‐167. IEEE, 2014.
27. Kaur K, Kaur A. Cloud era in mobile application testing. in computing for sustainable global development (INDIACom), 2016 3rd international conference on, pp. 1057‐1060. IEEE, 2016.
28. Tao C, Gao J, Li B. Cloud‐based infrastructure for mobile testing as a service. In Advanced Cloud and Big Data, 2015 Third International Conference on, pp. 133‐140. IEEE, 2015.
29. Zhang S, Pi B. Mobile functional test on TaaS environment. in service‐oriented system engineering (SOSE), 2015 IEEE symposium on, pp. 315‐320. IEEE, 2015.
30. Mahmood R, Esfahani N, Kacem T, Mirzaei N, Malek S, Stavrou A. A whitebox approach for automated security testing of android applications on the cloud. in automation of software test (AST), 2012 7th international workshop on, pp. 22‐28. IEEE, 2012.
31. Zhang T, Gao J, Cheng J, Uehara T. Compatibility testing service for mobile applications. in service‐oriented system engineering (SOSE), 2015 IEEE symposium on, pp. 179‐186. IEEE, 2015.
32. Grønli T‐M, Ghinea G. Meeting quality standards for mobile application development in businesses: a framework for cross‐platform testing. in system sciences (HICSS), 2016 49th Hawaii International Conference on, pp. 5711‐5720. IEEE, 2016.
33. Starov O, Vilkomir S. Integrated TaaS platform for mobile development: architecture solutions. in automation of software test (AST), 2013 8 th international workshop on, pp. 1‐7. IEEE, 2013.
34. Versluis G. Creating and running tests with xamarin test cloud. In Xamarin Continuous Integration and Delivery. Berkeley, CA: Apress; 2017:71‐91.
35. <http://tools.perfectomobile.com/>
36. Li A, Qin Z, Chen M, Liu J. ADAutomation: an activity diagram based automated GUI testing framework for smartphone applications. in software security and reliability, 2014 eighth international conference on, pp. 68‐77. IEEE, 2014.
37. Anbunathan R, Basu A. Automatic test generation from UML sequence diagrams for android mobiles. International Journal of Applied Engineering Research. 2016;11(7):4961‐4979.
38. Ali A, Badr N. Performance testing as a service for web applications. in intelligent computing and information systems (ICICIS), 2015 IEEE seventh international conference on, pp. 356‐361. IEEE, 2015.
39. Zheng Y, Cai L, Huang S, Wang Z. VM scheduling strategies based on artificial intelligence in cloud testing. in Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014 15th IEEE/ACIS International Conference on, pp. 1‐7. IEEE, 2014.
40. Liu C‐H, Chen S‐L. Evaluation of cloud testing strategies based on task decomposition and allocation for improving test efficiency. in Applied System Innovation (ICASI), 2016 International Conference on, pp. 1‐4. IEEE, 2016.
41. Paradigm, Visual. Visual paradigm for UML‐UML tool for software application Development 2013.
42. Janicki M, Katara M, Pääkkönen T. Obstacles and opportunities in deploying model‐based GUI testing of mobile software: a survey. Software Testing, Verification and Reliability. 2012;22(5):313‐341.
43. Cartaxo EG, Neto FGO, Machado PDL. Test case generation by means of UML sequence diagrams and labeled transition systems. In Systems, Man and Cybernetics, 2007. ISIC. IEEE International Conference on, pp. 1292‐1297. IEEE, 2007.
44. Berenbach B, Paulish D, Kazmeier J, Rudorfer A. Software & systems requirements engineering: in practice. McGraw‐Hill. In: Inc. ; 2009.
45. Gudmundsson V, Lindvall M, Aceto L, Bergthorsson J, Ganesan D. Model‐based testing of mobile systems‐‐an empirical study on QuizUp android app.
46. arXiv Preprint arXiv:1606.00503 2016.
47. Sipser M. Introduction to the Theory of Computation. Vol. 2. Boston: Thomson course Dent Tech; 2006.
48. Heineman GT, Pollice G, Selkow S. Algorithms in a nutshell: a practical guide. O'Reilly Media, Inc., 2016.
49. Hirschberg DS. Algorithms for the longest common subsequence problem. Journal of the ACM (JACM). 1977;24(4):664‐675.
50. APPIUM, URL http://appium.io/index.html? lang=en
51. Singh S, Gadgil R, Chudgor A. Automated testing of mobile applications using scripting technique: a study on Appium. International Journal of Current Engineering and Technology (IJCET). 2014;4(5):3627‐3630.
52. APPIUM, URL <http://appium.io/index.html?lang=en>
53. http://robotium.com 53. http://calaba.sh/
54. Rosenblum M. Vmwares virtual platform. In Proceedings of hot chips, vol. 1999, pp. 185‐196; 1999.
55. <https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html>
56. Agrawal S, Chaudhuri S, Kollar L, Marathe A, Narasayya V, Syamala M. Database tuning advisor for microsoft sql server 2005. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pp. 930‐932. ACM, 2005.
57. MonkeyRunner, http://developer.android.com/tools/help/monkeyrunner\_concepts.html
58. <https://apkpure.com/simple-notepad/org.mightyfrog.android.simplenotepad>
59. <https://apkpure.com/clock/com.google.android.deskclock>

附录A

测试活动图下的应用程序



**图A.1** 简单记事本应用程序活动图



图A.2 时钟应用程序活动图