文章编号:1671-4598(2022)05-0030-07 DOI:10.16526/j.cnki.11—4762/tp.2022.05.006 中图分类号:TP311.5 文献标识码:A

基于模型驱动的 Web 应用自动化测试平台设计与应用

羊铃霞[1]，陈元松[1]，仵林博[2]，尚小虎[2]，陈立伟[1]
(1. 西南科技大学 计算机科学与技术学院，四川 绵阳 621010；
2. 中国工程物理研究院 计算机应用研究所，四川 绵阳 621999)

摘要：为了提高 Web 应用的测试效率和测试覆盖率，保证 Web 应用软件的质量，设计了基于模型驱动的 Web 应用自动化测试平台；该平台结合了基于 UML 模型的测试用例生成、基于关键字驱动思想的框架设计和复杂多层的自动化测试框架，实现了测试用例自动设计生成及测试自动化执行，增加了测试脚本的复用性，显著提高了测试效率和测试覆盖率；最后，给出应用实例，并与现有的测试方法和平台进行对比，突出本平台的可行性和应用价值。

关键词：自动化测试；模型驱动；测试用例；关键字驱动；Web 应用

Design and Application Based on Model-driven Automatic Test Platform for Web Application

YANG Lingxia[1], CHEN Yuansong[1], WU Linbo[2], SHANG Xiaohu[2], CHEN Liwei[1]
(1. School of Computer Science and Technology, Southwest University of Science and Technology, Mianyang 621010, China;
2. Institute of Computer Application, China Academy of Engineering Physics, Mianyang 621999, China)

Abstract: In order to improve the test efficiency and coverage of Web application and ensure the quality of Web application software, based on the model driven, an automatic test platform for Web application is designed. The platform combines the test case generation based on UML model, based on keyword-driven idea and the complex multi-layer automated testing framework, the framework design realizes the automatic design and the test case generation and the automatic execution, increases the reusability of test scripts, and significantly improves test efficiency and test coverage. Finally, an application example is given, and compared with the existing method and the platform, the feasibility and application value of the platform is emphasized.

Keywords: auto-testing; model-driven; test case; keyword-driven; web application

0 引言

近年来，在计算机技术和互联网技术飞快发展之下，B/S 架构的 Web 应用因其操作简单、快速、无需安装等特点，占据了软件市场的大量份额。随着 Web 应用的快速发展，Web 应用的架构层次越来越复杂、质量要求越来越高、测试难度也越来越大。Web 应用在测试过程中需要花费大量的时间编写测试用例，执行测试用例，并且在新版本发布时，回归测试也将花费大量的物力人力，在这样的形式下，需要一个可以提高测试效率、减少人力物力的 Web 应用自动化测试平台。

目前，国内外有许多针对基于模型的测试用例自动生成和自动化测试执行的研究，文献 [1] 提出了一个基于模型的测试平台，称为 ModCon，依赖于用户指定的模型来定义测试预言，指导测试生成和度量测试充足率，主要用于支持无权限和有权限的区块链平台。文献 [2] 引入了一种新的基于模型的测试方法，根据需求和故障模式对测试用例进行排序，在基于模型的测试方法中使用了失败模式和效果分析方法，自动生成一组成对的测试用例，利用优先级编号来排序测试用例。文献 [3] 引入了一种新的基于模型的 IFML UI 元素测试方法，使用形式化模型提供完整的导航测试，通过生成状态转换矩阵和详细的 UI 测试用例文档，将 IFML 模型转换为必要的 UI 测试工件，实现了基于模型的用户界面测试用例 (MBUITC) 生成器工具。文献 [4] 研究活动图的建立规则，对循环结构和并发结构处理，提出了 UML 活动图和遗传算法相结合的

收稿日期:2021-10-26； **修回日期**:2021-12-06。
基金项目:国家自然科学基金青年基金项目(61903348)。
作者简介:羊铃霞(1995-)，女，四川绵阳人，硕士研究生，主要从事软件测试方向的研究。
陈立伟(1974-)，男，重庆永川人，博士，教授，硕士研究生导师，主要从事信息处理、模式识别方向的研究。
通讯作者:陈元松(1977-)，女，四川绵阳人，硕士，讲师，主要从事软件测试方向的研究。
引用格式:羊铃霞，陈元松，仵林博，等.基于模型驱动的 Web 应用自动化测试平台设计与应用[J].计算机测量与控制，2022，30(5)：30-36.

测试用例生成方法。文献［5］提出了一种基序列图和状态图关联关系生成测试用例的方法，该方法有效发现软件在处理多对象交互情景下的缺陷。文献［6］针对目前 Web 测试费时费力，提出了一种基于 WSDL 文档和形式化模型树 Web 服务操作测试用例的自动生成方法。文献［7］针对 Web 应用需求频繁更改问题，研究了基于低耦合的 Web 自动化测试框架，实现数据模块、业务逻辑和结果显示模块相分离。文献［8］针对回归测试占用大量测试人力资源的现象，设计并实现了一种基于 Selenium 与 Unittest 的 Web 自动化测试框架。

在上述自动化测试技术研究中，大多数工具只针对某个方面的研究，如只关注测试用例自动生成或自动化的测试执行，很少从测试需求分析、测试用例设计生成、测试用例执行、测试报告生成等软件测试的全流程给出研究方案，并且现在的工具没有真正意义上的能够普适各种 Web 应用和全自动化的结论性成果。本文针对软件测试整体流程，设计了一套高适应性的模型驱动的 Web 应用自动化测试平台，该平台实现了测试用例自动设计、测试脚本自动生成、自动化的测试执行以及测试报告的生成，在保证 Web 应用软件质量的同时，显著提高了测试效率和测试覆盖率。

1　自动化测试平台概述

该自动化测试平台主要包括被测系统建模、测试策略制定，测试用例生成，测试执行等功能。该平台是一个以 UML 模型为基础的自动化测试的 Web 平台，其中用户需求、测试策略以及测试用例均用 UML 模型进行表示，整个测试过程都依赖 UML 模型。

1.1　自动化测试工作流程

模型驱动的自动化测试是先使用 UML 状态图模型形式化地描述将要被测试的 Web 应用的业务流程，再利用 UML 状态图模型的状态和迁移动作信息来自动产生测试用例和测试脚本，然后，通过自动执行系统执行生成的测试用例和测试脚本，最后，根据测试执行结果生成最终的测试报告。模型驱动的自动化测试组成、工作过程如图 1 所示。

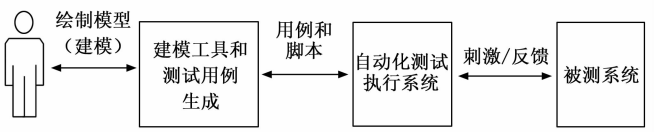

图 1　模型驱动的自动化测试工作过程

1.2　自动化测试平台构成

本文所述自动化测试平台主要由服务器端和客户端组成，其中主要的应用功能模块可分为 3 个组成部分。

1.2.1　测试模型

测试设计人员通过需求分析，利用 UML 用例图建立被测软件的需求模型，清晰表达用户的测试需求和测试重点，确定大致的测试策略和测试路径；采用基于 UML 状态图的建模方式来对被测应用的业务逻辑行为进行抽象的描述，帮助梳理被测试应用的业务逻辑，表达出被测对象的操作动作和状态，并为其绑定关键字；采用“业务模型＋数据驱动”的模式对特定场景中输入数据和操作数据根据一些测试方法（如等价类划分、边界值等）编写测试数据进行建模，再根据模型生成不同组的测试数据；最后在对应的模型中配置一组或几组测试数据以及一些约束条件，完成整个被测应用的业务行为模型。

根据上述，模型驱动的自动化测试平台中的模型一般而言分为 3 种：用例模型，数据模型，行为模型。其中，数据模型和行为模型又被统称为状态图模型。

用例模型：用于对用户需求（如界面迁移图，测试策略，用例说明，原型系统等）的表述。建立该模型主要为了：清晰直观地捕捉用户需求；迅速暴露测试计划制定时可能的缺漏点；建立从需求到结果的双向可追溯链条，使测试活动呈现得更加明确；使得测试结果覆盖的内容能够直观的与需求联系在一起。

状态图模型：主要用于对被测对象的期待的行为进行描述。数据模型和行为模型分别描述了被测对象在被期待的行为进行中所需的数据和动作或状态。利用该模型，能够直接表述出期望的被测对象的动作及状态；填入测试脚本，使得动作、状态与行为能够对应；可以自动生成测试用例，替代了人工的测试用例设计和编写，提高了工作效率。

1.2.2　测试用例及测试脚本

测试用例及测试脚本是通过基于模型的分析方法自动生成的，主要根据基于状态图在状态和迁移的拓扑图中枚举出所有的测试路径，综合列举出所有可能的操作与对应操作数据参数，并对数据参数取值进行组合产生测试用例。同时，为了生成自动化测试执行的测试脚本，需要编写相应被测系统的操作关键字实现业务逻辑封装，即将人对被测应用行为的操作、对界面元素的操作、输入输出操作，通过编写操作关键字的形式进行封装。通过为 UML 状态图中各个迁移动作和状态全部绑定编写的关键字，再结合枚举出的测试路径就可以生成可执行的测试脚本。最后，调用自动化测试执行系统进行执行，即平台生成的用例就可以利用测试脚本实现全部自动化运行。

1.2.3　自动化测试执行系统

自动化测试执行系统 ATE（auto test executing），该部分是对各个应用领域的底层封装，主要是利用关键字驱动的思想对 Selenium 的二次开发再封装，例如，把 Selenium 对浏览器驱动和对 Web 应用元素定位、操作原本的方法通过编写 Python 脚本设计关键字的形式进行封装。利用二次封装增加了测试脚本的可复用性，降低了测试脚本的编写难度。最重要的是可以解析生成的测试脚本使得被测系统像被人工操作一样的自动运行，该平台生成的测试用例就可以调用 ATE 进行 7×24 小时全自动运行，显著地提高了测试效率。

2 自动化测试平台设计

模型驱动的 Web 应用自动化测试特点是根据被测系统模型及其派生模型来产生测试用例和测试脚本，进行测试自动执行，以及测试结果展示[9]。在基于模型的测试中，被测应用的测试模型和基于测试模型生成的测试用例是抽象的，并且是独立于平台，可重复使用的。测试执行时通过对测试执行平台的动态配置自动产生实例化的可执行的测试脚本。本平台是将测试模型、测试用例平台和自动化测试执行平台分离实现，通过适配器模块连接模型工具和执行平台，降低 Web 应用的异构性和动态性所带来的测试复杂性[10]。因此，模型驱动的自动化测试主要用到的技术就是基于 UML 模型的测试用例生成、基于关键字驱动思想的框架设计和复杂多层的自动化测试框架。

2.1 基于 UML 模型的测试用例生成

基于 UML 模型[11-15]的测试用例生成主要是按照被测系统的业务流程和需求规格说明对整个被测系统进行 UML 状态建模，然后根据建立的状态图的状态和迁移将其转化为有向图，最后通过测试路径生成方法、测试覆盖生成策略结合一些测试数据得到相应的测试用例集。测试用例生成基本流程如图 2 所示。

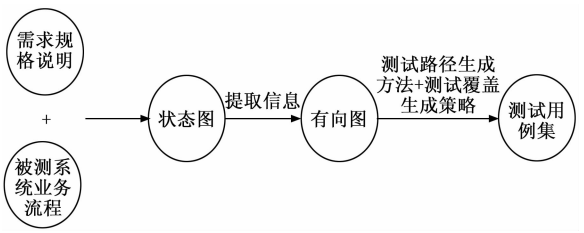

图 2 测试用例生成流程图

2.1.1 测试路径生成方法

测试模型本身是通过“图”（拓扑图）的实例体现，测试用例生成算法则可以基于图论的各种经典算法来达成对图中各种路径的枚举，简单的枚举将可能造成测试用例数量爆炸和测试用例数量过少两种极端的结果，为防止造成这样的结果，将基于风险的测试方法融入到了测试用例生成算法之中。测试用例生成算法不仅仅在图中枚举路径，而且是根据图中的优先级（与软件需求的关系密切程度）来优先生成排序。将风险高的，更为基础的测试用例排列在生成的测试用例集合前面，而把风险相对较低的排列在稍后的位置上，从而在测试覆盖和资源消耗之间找到一个平衡。本平台使用了深度优先遍历算法对行为模型转换的有向图进行枚举遍历得到所有的测试路径以及使用软件产品风险与生成算法优先级对测试路径进行排序选择。

1）基于深度优先遍历的测试路径生成。

测试路径生成采用深度优先遍历算法获取有向图开始点和结束点两个之间的所有路径。有向图中各个边通过邻接矩阵方式进行存储。

基于深度优先遍历的测试路径生成算法：

```
输入：一个有向图边的邻接矩阵 matrix 和点集合 vertex；
输出：路径集合 P={P1(v0,…,vm),P2,…,Pn}；
//使用深度优先遍历找到所有路径；
初始化 P={};//定义集合为空；
Function countPathNumber(){}; //计算路径分支数；
Function getPaths(){；
For i －> countPathNumber()：
path ={} ；//用于保存遍历过的点；
DFS(0,path)；//调用深度优先遍历算法；
P. add(path)；//将遍历过的路径保存到路径集合；
End for；
}
//返回所有的路径集合
Return P
End function
```

2）软件产品风险与生成算法优先级概要。

产品风险对应的名词解释如下。

Trunk：直接反映需求对应的软件行为的状态节点。

Relation：与需求产生关联的软件行为的状态节点。

Normal：系统中与对应需求相关性低或无已知联系的状态节点。

从基于风险的测试策略角度来看，trunk 对应的是功能是否正常工作的直接风险；relation 对应的是功能稳定性，互操作性相关质量属性的风险；normal 对应的是功能可能相关的低级风险。

软件产品或者系统中，与功能需求直接对应的软件行为对应了最高优先级的风险，因为如果基本功能都无法运转，则软件产品完全失去了价值；功能的稳定性，性能，在复杂场景下的互操作性等质量属性则是在满足基本功能行为的基础上，更高的质量要求，即这些要求对应的软件行为和处理的风险相对于基本功能而言比较低一些；非功能需求对应的软件行为处于更低的风险级别。因此，根据上述可以人为的设置各种定量的风险级别计算，这里采用 3 级基本风险：Trunk（主干），Relation（关联），Normal（普通），并由这 3 种基本风险级别的组合来体现更多更丰富的产品风险级别。

由于模型图中的任意节点都可以根据实际系统的情况标记为“Trunk”，“Relation”，“Normal”的任意一种，故此，在模型图中生成的测试用例所对应的拓扑图中路径所包含的节点序列可以是多种节点集合的情况，比如：{起始节点，Normal 节点集合一，Trunk 节点集合一，Normal 节点集合二，Relation 节点集合一，Trunk 节点集合二，结束节点}。所以，采用“模式”的方法来描述测试用例生算法将符合什么样的模式。上例中的模式可以总结为：Start，Normal，Trunk，Normal，Relation，Trunk，End。如果在生成算法中仅关注关键模式，而忽略次要模式，则上例可归纳为：

Start－>Trunk－>Relation－>Trunk－>End

上述模式中，我们忽略了序列中的 Normal 节点集合。

原因是，在实际的优先级设置时，与某个需求关联的节点可能与拓扑图中的起始节点不相邻。故此，在生成时，将集中按模式关注点生成子路径集合，然后再在起始节点与子路径的第一个节点之间寻找一条最短路，以及在子路径的结束节点到拓扑图的终点节点之间寻找一条最短路径。这样可能会造成一种情况：即在起始节点到子路径以及子路径到终点节点之间的路径中可能包含 Trunk，Relation，Normal 等各种可能的优先级，实际上“－＞”这个符号将代表任意在模式中不关注的路径内容，其内容在算法中不予关注，故此其可能重复。

综上，将生成的测试路径集合结合基于风险的测试方法就可以得到合适数量的测试路径集合。

2.1.2　测试覆盖生成策略

测试用例的自动生成需要按照需求制定相应的测试覆盖生成策略，自动生成满足测试需求的测试用例集，其中基于 UML 状态图模型的测试用例生成的核心就是测试覆盖生成策略的制定，本平台主要用到两个测试覆盖生成策略：状态全覆盖生成策略和主辅功能优先覆盖生成策略。

状态全覆盖生成策略本质上是对 UML 状态图所建立的被测系统业务模型生成的所有测试场景，达到最全面的覆盖。在此覆盖下，将获取到所有符合参数定义规约的测试路径，因此，在这种测试覆盖生成策略下，将产生非常庞大的测试用例集合。这种策略适用于早期初次的测试来确保满足覆盖。

主辅功能优先覆盖生成策略本质上是 UML 状态图所建立的被测系统业务模型生成的满足需求的测试场景，达到主要功能和场景的覆盖。在此覆盖下，将获取到与需求有明确对应关系的路径和已知可能与需求发生交互影响的路径和测试场景，最后再添加上影响不大或影响未知的行为路径。这种策略适用于按照一定需求来测试主要功能与场景的测试。

同时，为了控制测试用例集的数量，该平台还配置了最大测试用例数，状态图环大小，环次数以及用例的最大步长等参数来限制测试用例的数量。

2.2　基于关键字驱动思想的框架设计

关键字驱动[16-18]是将业务逻辑、数据和脚本分离，提高代码的可重用性，提高脚本的可维护性的思想。关键字驱动测试的核心就是对关键字进行设计与封装，传统的关键字封装就是对浏览器的操作、被测对象、定位方式和值等情况进行封装，再结合单元测试框架 Unittest 和 Pytest 搭建相应的测试框架[19]。

本文针对被测应用的常用操作和建立的 UML 状态模型的业务流程，将关键字的思想应用于平台中，主要是对 Web 应用的常用操作和根据被测应用建立 UML 状态模型的迁移和状态，利用关键字思想设计了相应的浏览器驱动关键字、数据获取关键字、断言关键字和业务流程关键字等。同时，将设计的业务流程关键字绑定在状态模型图的状态和迁移上，通过 UML 状态模型图和绑定的关键字就可以自动生成关键字的测试脚本。最后，根据生成的测试脚本驱动被测应用进行相应的操作，实现被测应用的自动化测试。关键字驱动框架如图 3 所示。

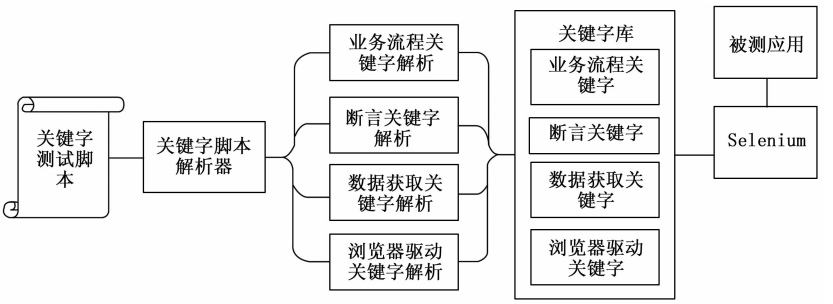

图 3　关键字驱动架构

2.3　复杂多层的自动化测试框架

分层测试框架[20-22]有助于测试设计和测试开发解耦，提高一些模块的复用性以及测试的覆盖率。本平台是基于 Selenium 的 Web 应用自动化测试的扩展与改进框架，主要分为客户端和服务器端，并从测试脚本层、测试执行层、业务展示层 3 个层面出发。整个测试框架的整体设计如图 4 所示。

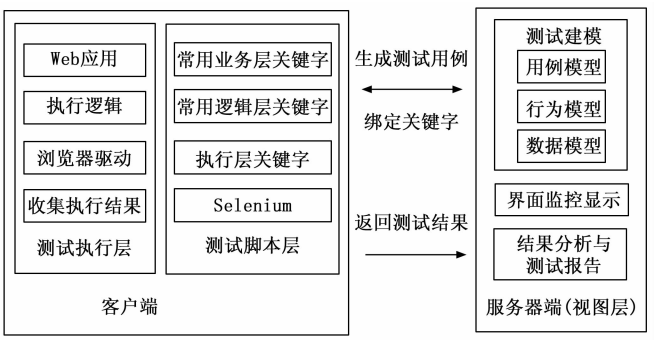

图 4　测试框架图

测试脚本层主要是对 Selenium 操作进行二次封装，该层使用关键字驱动的自动化测试技术将用户的执行操作和业务逻辑封装成关键字，为了降低脚本之间的耦合性，增加脚本的灵活性，根据关键字所在层次分为执行层关键字、常用逻辑层关键字和常用业务层关键字。

测试执行层主要是通过测试脚本层的脚本运行调用相应浏览器驱动和相应的执行逻辑对 Web 应用进行自动化操作，记录其操作日志和收集执行结果，并将其反馈给服务器端。

业务展示层主要是对被测应用进行建模、测试监控和测试结果分析展示，其中测试建模是通过用例模型对用户需求进行建模；通过行为模型对被测应用业务流程进行梳理，描述被测对象期待的行为；通过数据模型建立被测对象行为所用到的数据。测试监控对测试过程路径进行实时监控，测试结果分析展示就是对测试用例和测试点的通过

情况进行展示。

综上，通过复杂多层的测试框架将测试过程细化，并且实现测试数据与测试逻辑分离，降低了测试脚本的耦合性，提高了测试脚本的复用性，达到了一定的灵活性。

3 应用验证

为了覆盖上述的测试方法，验证平台的可行性和有效性，本文主要从 3 个方面来对该自动化测试平台进行验证。

1）选择了一个在线教育平台作为被测对象，进行测试验证；

2）利用多维度测试覆盖率和测试时间成本对该平台进行分析，并与 Spec Explorer 工具进行对比；

3）给出该平台做过的 Web 应用的自动化测试实验和数据。

3.1 实验案例

根据上述设计实现了自动化测试平台，并通过一个基于 B/S 架构的在线教育学生端系统进行应用验证，主要通过模型建立、关键字设计、用例自动生成和测试用例自动执行等几个方面进行应用验证。

1）梳理整个被测系统的业务逻辑流程，如登录、搜索、选课、直播等功能，并利用自动化测试平台的 UML 状态图建立其被测系统的行为模型，如图 5 所示。

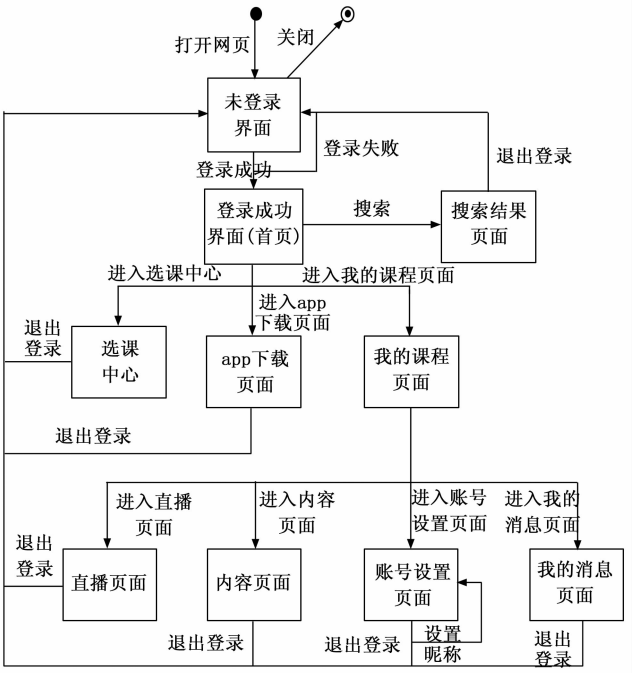

图 5 行为模型

2）根据建立的行为模型状态图的每个状态和迁移动作进行定义相应的关键字，如打开网页、登录成功、退出登录等关键字，并将这些关键字与行为模型的元素进行对应绑定，如图 6 所示。

3）在主要的行为模型建立完成之后，将使用 UML 用例图根据测试需求建立相应的需求模型，再设置行为模型


脚本输入

	状态类型	状态名称	脚本	添加脚本
1	起点	start		插入关键字
2	终点	end		插入关键字
3	元素	搜索结果页面	Education.searchResults()	插入关键字
4	元素	我的课程页面	Education.wdkcCheck()	插入关键字
5	元素	未登录界面	Education.unlogincheck()	插入关键字
6	元素	登录成功界面（首页）	Education.loginSuccessCh	插入关键字
7	元素	app下载页面	Education.appCheck()	插入关键字
8	元素	直播页面	Education.zbymCheck()	插入关键字
9	元素	内容页面	Education.nrymCheck()	插入关键字
10	元素	账号设置页面	Education.zhszCheck()	插入关键字
11	元素	我的消息页面	Education.wdxxCheck()	插入关键字
12	连接线	打开网页	Education.open(url)	插入关键字
13	连接线	关闭	Education.close()	插入关键字
14	连接线	登录成功	Education.loginSuccess(us	插入关键字
15	连接线	进入app下载页面	Education.appJump()	插入关键字
16	连接线	搜索	Education.search(searchVi	插入关键字
17	连接线	进入我的课程页面	Education.wdkcJump()	插入关键字
18	连接线	设置昵称	Education.zhszSave()	插入关键字
19	连接线	进入账号设置页面	Education.zhszJump()	插入关键字
20	连接线	进入我的消息页面	Education.wdxxJump()	插入关键字

图 6 关键字

中的主功能和辅功能的状态，并与需求模型进行关联，然后配置相应的测试策略，设置待生成的测试用例总数为“100”，生成所有大小的环为“是”，环允许包含的节点数为“0”，环重复的次数为“1”，单个测试用例最大步骤数为“100”，生成算法为“全覆盖”，生成测试用例集合。

4）在测试用例集合生成之后，进入测试用例执行界面，选择需要执行的用例集合，点击“执行”，选择执行设备，开始自动执行测试用例，自动化测试平台对结果进行反馈，如图 7 所示，测试反馈结果中，一共选择了 19 个用例进行执行，包含了 273 个测试点，用例通过为 17 个，未通过 2 个，测试点通过为 265 个，未通过为 8 个。

3.2 实验评估

为了验证本文测试方法和测试平台的研究成果，选择了 Spec Explorer 工具进行对比实验，主要从测试覆盖率和测试流程时间两个方面进行对比分析。

3.2.1 测试覆盖率计算

基于文献［7］提出的多维度软件测试覆盖率的评估概念：

$$覆盖率 = \left(\frac{至少被执行一次的\ item\ 数}{item\ 总数}\right)\times 100\% \quad (1)$$

公式（1）中，需要从不同角度对 item 进行统计，就是考虑不同侧重点的测试覆盖率情况。

根据文献［7］的综合测试覆盖率和综合满意度的概念：

$$综合覆盖率 = \left(\frac{\sum_{i=1}^{n} C_i P_i}{\sum_{i=1}^{n} P_i}\right)\times 100\% \quad (2)$$

公式（2）表示综合覆盖率，n 表示测试覆盖率的维度，C_i 表示第 i 维度的测试覆盖率，这是由式（1）得到，其中 $0\leqslant C_i\leqslant 1$，$1\leqslant i\leqslant n$，$P_i$ 表示该维度覆盖率的权重。

$$综合满意度 = \left(\frac{\sum_{i=1}^{n} (C_i/E_i)\times P_j}{\sum_{i=1}^{n} P_i}\right)\times 100\% \quad (3)$$

公式（3）表示测试的综合满意度，E_i 表示第 i 维度的测试覆盖率的期望值，其中 $0\leqslant E_i\leqslant 1$，$1\leqslant i\leqslant n$。

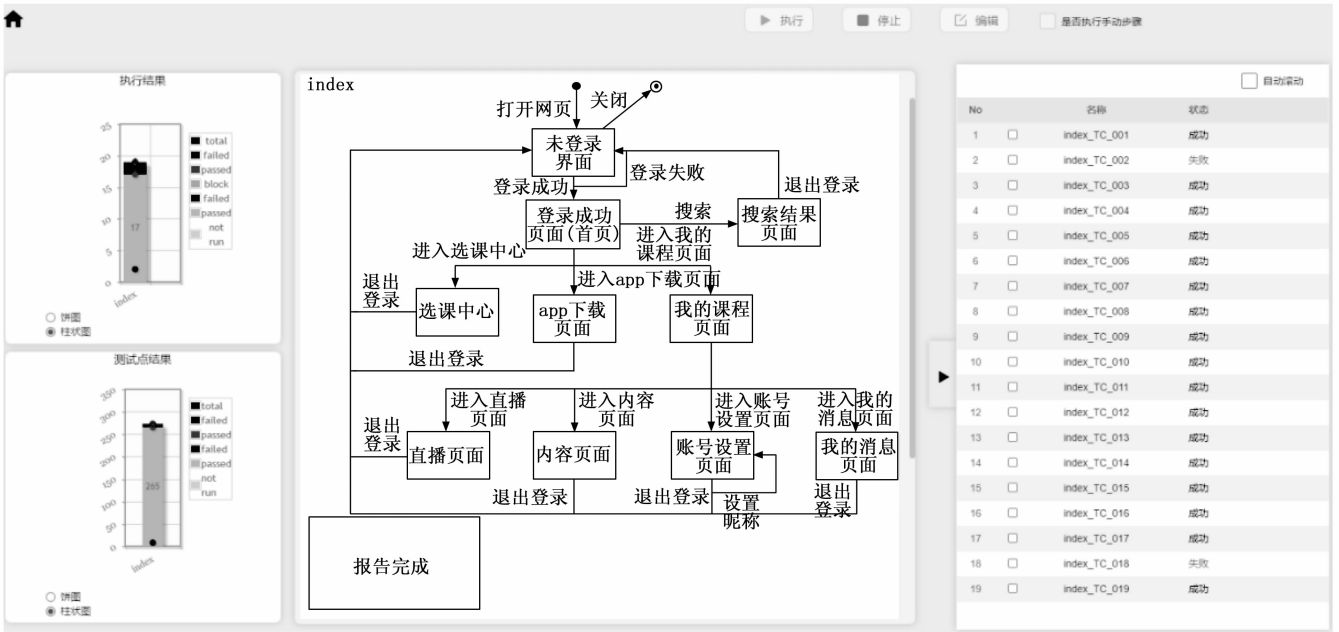

图 7　测试报告

3.2.2　实验对比

实验挑选 5 个从未用过本文平台和 Spec Explorer 工具的测试人员，给他们一段时间熟悉这两个工具，对两个工具熟悉之后，这 5 个测试人员分别使用这两个工具对实验案例进行测试，统计出测试过程中每个环节所需要的时间，然后求出测试过程中每个环节的平均时间。

对上述案例进行测试所需平均时间成本对比如表 1 所示。

表 1　测试流程所需时间对比

测试流程	本文平台/min	Spec Explorer/min
建立测试模型	39	58
编写测试数据	23	20
编写关键字	132	0
编写测试脚本	0	169
测试执行	37	56

实验过程中主要是对 web 应用功能创建行为模型生成测试用例集，实验对生成的测试用例集中满足需求覆盖率情况进行对比分析，主要从功能点覆盖率、页面提示覆盖率、流程覆盖率、功能组合覆盖等情况根据公式（1）计算覆盖率，对比情况如表 2 所示。

表 2　覆盖率情况对比

	覆盖率 C		期望 E	权重 P
	本文平台	Spec Explorer		
功能覆盖率/%	100	98.20	100	5
页面提示覆盖率/%	94.54	78.65	75	2
流程覆盖率/%	98.46	96.45	100	3
功能组合覆盖率/%	98.78	70.60	70	1

根据表 2、公式（2）和公式（3）分别计算出综合覆盖率和综合满意度如表 3 所示。

表 3　综合覆盖率和满意度对比

	综合覆盖率/%	满意度/%
本文平台	98.48	108.05
Spec Explorer	91.66	99.18

3.3　综合案例

为了验证本平台的适应性，选取 5 个不同大小，不同领域的 Web 应用进行了全流程自动化测试，根据 Web 应用实际情况建立模型，设计关键字，生成测试用例，最后生成测试报告，测试过程数据如表 4 所示。

表 4　Web 应用自动化测试数据

Web 应用	关键字个数	测试用例数	测试点数	缺陷数	执行时间
寻物管理系统	35	20	242	3	32 min
文化资料管理系统	64	67	712	10	90 min
在线教育系统	58	40	712	15	88 min
华信达智能垃圾回收平台	45	21	197	5	30 min
电子会议系统	93	1 000	22 992	12	40 h

4　结束语

本文使用了基于 UML 模型的测试用例生成方法、基于关键字驱动思想的框架设计和复杂多层的自动化测试框架，搭建了模型驱动的自动化测试平台，并通过相应的实验验证。该平台通过复杂多层的自动化测试框架降低了测试框架的耦合性，增加了测试脚本的灵活性和复用性。全流程自动化执行，提高了测试人员的测试效率，增加了测试覆盖率，并适用于各类的 web 应用的自动化测试。

参考文献：

[1] LIU Y，LI Y，LIN S W，et al. ModCon：A model-based testing platform for smart contracts [C] //Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2020：1601－1605.

[2] MERIEM A，ABDELAZIZ M. A Methodology to do Model-Based Testing using FMEA [C] //Proceedings of the 2nd International Conference on Networking，Information Systems & Security. 2019：1－11.

[3] YOUSAF N，AZAM F，BUTT W H，et al. Automated model-based test case generation for Web user interfaces (WUI) from interaction flow modeling language (IFML) models [J]. IEEE Access，2019，7：67331－67354.

[4] 梁卓杰. 测试用例自动生成算法设计及自动化测试平台构建 [D]. 北京：北京交通大学，2019.

[5] 刘艳平，李海浩. 基于序列图和状态图的软件测试用例生成方法 [J]. 电子设计工程，2019，27 (24)：167－170，175.

[6] 侯 俊，周 红，马春燕，等. 面向 WEB 服务的测试用例自动化生成方法 [J]. 西北工业大学学报，2018，36 (1)：149－155.

[7] 边耐政，张 琳. 一种基于 Selenium 的 Web 自动化测试低耦合框架 [J]. 计算机应用与软件，2014，31 (8)：13－16，37.

[8] 单攀攀. 一种基于 Selenium 与 Unittest 的 Web 自动化测试框架 [J]. 信息技术与网络安全，2021，40 (9)：77－80.

[9] 缪淮扣，陈圣波，曾红卫. 基于模型的 Web 应用测试 [J]. 计算机学报，2011，34 (6)：1012－1028.

[10] 曾红卫. Web 应用的验证与测试方法研究 [D]. 上海：上海大学，2008.

[11] 张毅伟，贲可荣. 基于状态图测试的迁移路径生成方法 [J]. 计算机科学与探索，2019，13 (6)：961－972.

[12] 陈小林. 基于 UML 模型的测试用例自动生成综述 [J]. 现代计算机 (专业版)，2018 (7)：61－65.

[13] 张 建. 基于模型驱动的自动化测试平台相关技术研究 [D]. 杭州：杭州电子科技大学，2017.

[14] 尹建月，周 萌，陈升来. 基于 UML 活动图的测试用例生成方法设计 [J]. 信息化研究，2014，40 (3)：28－32.

[15] 齐 磊. 基于 UML 的 Web 应用测试方法的研究与应用 [D]. 北京：北京工业大学，2012.

[16] 王 军，孟凡鹏. 基于关键字驱动的自动化测试研究与实现 [J]. 计算机工程与设计，2012，33 (9)：3652－3656.

[17] 康新欣. 基于关键字驱动的移动端自动化测试平台的设计与实现 [D]. 北京：北京工业大学，2019.

[18] 章鸽鸽. 基于关键字驱动的自动化测试工具的设计与实现 [D]. 合肥：安徽大学，2018.

[19] 孙婧鑫. 基于 Selenium 的 Web 功能自动化测试框架的设计与实现 [D]. 西安：西安石油大学，2021.

[20] 杨 静. 自动化测试平台的设计与实现 [D]. 北京：北京交通大学，2020.

[21] 陈江勇，许 力，张 辉，等. Web 自动化测试框架的设计与实现 [J]. 福建师范大学学报 (自然科学版)，2013，29 (4)：39－45.

[22] 万 琳，廖飞雄. 一种分层结构测试脚本技术 [J]. 计算机系统应用，2011，20 (7)：141－145.

(上接第 29 页)

[4] 姚斯可，刘 龑. 软件通信体系结构 CORBA 中间件应用及指标评测 [J]. 航空电子技术，2020 (2)：37－43.

[5] 朱 娇，祝颂东，阮轶杰. 外军软件通信体系结构规范 [J]. 电子技术与软件工程，2020 (7)：18－20.

[6] 赵 蕾. SCA 异构平台系统波形部署技术研究与实现 [J]. 通信技术，2020，53 (5)：1157－1162.

[7] 王彦刚，吕遵明，万留进. 基于 SCA 规范的硬件抽象层应用程序接口分析 [J]. 计算机应用，2014，34 (S2)：219－223.

[8] 邵 龙. 多总线结构 MHAL 路由方法 [J]. 电讯技术，2020，60 (10)：1228－1232.

[9] 钟 瑜. 通用信号处理中的硬件抽象层连接设计 [J]. 电讯技术，2011，51 (6)：46－50.

[10] 崔晓鹏，胡中豫，张 豪. SCA 中 CORBA 与硬件抽象层技术研究 [J]. 现代电子技术，2011，34 (6)：32－35.

[11] 林 婧，王 宏，方 炜. 软件无线电的研究现状综述 [J]. 计算机测量与控制，2011，19 (10)：2332－2350.

[12] 张 迪，韩 爽. 基于软件无线电平台的中频信号处理系统设计 [J]. 计算机测量与控制，2019，27 (11)：220－223.

[13] 高英明，陈建云，唐银银. 基于 SOC 架构软件无线电平台的低轨卫星载荷接收信号模拟技术研究 [J]. 计算机测量与控制，2015，23 (11)：3724－3730.

[14] 郭 敏，庄信武，王向东，等. 基于软件无线电架构的手持式频谱仪硬件设计 [J]. 计算机测量与控制，2018，26 (1)：296－299.

[15] 高竞之，郭兴龙，姚金杰，等. 基于软件无线电的基带调制系统设计 [J]. 计算机测量与控制，2012，20 (12)：3289－3291.

[16] 邱 雅，郭东恩. 一种基于软件无线电的中频数字接收机的设计实现 [J]. 计算机测量与控制，2012，20 (4)：1070－1075.

[17] 习郑虎，年夫顺，樊晓腾，等. 基于 FPGA 的多制式基带信号发生器设计 [J]. 计算机测量与控制，2011，19 (10)：2584－2586.

[19] 石贱弟，赵小璞. 基于软件通信体系结构的 DSP 硬件抽象层研究与设计 [J]. 电子设计工程，2011，19 (17)：123－125.

[20] Joint Tactical Radio System (JTRS) Joint Program Office. Modem Hardware Abstraction Layer Application Program Interface [S]. MHAL API V2.13.2. USA：JTRS Joint Program Office，2013.