1. 绪论

本文先是讨论了测试的必要性以及自动化测试在软件开发中的重要性，再而介绍了BDD（行为驱动开发）的原理与支持BDD的自动化测试Cucumber框架，并着重介绍了基于Cucumber测试用例的全平台自动化测试web系统的设计。

软件测试是对项目给定需求的功能验证，是软件质量管理Quality Assurance（品质保证）的重要手段。1979年，Glenford Myers在《The Art of Software Testing》对软件测试作出了定义：“软件测试是为了发现错误而执行程序或系统的过程”。而基于TDD（测试驱动开发）的思想，本文对软件测试有了新的定义：软件测试是通过执行程序，发现系统漏洞与bug来进行对产品质量的验证，软件测试能修正软件的错误，或在测试中发现产品的非错误性不足以驱动开发改进。

软件测试具有发现产品质量问题并改善问题的功能，而在实际开发中，软件测试是不可或缺的，软件测试在当前对软件应用的高安全性、可用性、高质量性的需求当中，扮演着愈发重要的角色。

1.1研究背景与意义

随着应用软件的发展、测试难度的需求复杂化，自动化测试的理念逐渐被大多数企业所接受。并且在软件过程管理中，随着迭代式开发与敏捷开发等框架取代瀑布模型，测试的周期缩短，工作增加，需要对软件的多个迭代版本进行测试，并增加了许多重复性的测试劳动，自动化测试的优点是显而易见的。

软件测试主要由四部分组成：被测应用、环境、方式、测试人员。

完善的自动化流程可以让环境部署的时间大大减少，以自动执行一些重复但是必要的任务的方式减轻测试人员的负担，并且解决许多手工测试完成不了的工作（如并发访问等），所以标准的自动化测试软件服务对项目持续交付和持续测试意义深远，同时对于项目的质量检测、监测都是不可或缺的。

然而，自动化测试并不是万能的，自动化测试在相当一部分的环境下，具有极大的局限性。首先，自动化测试难以管理，在对系统进行全面的自动化测试当中，需要将许多的测试脚本综合管理。并且，自动化测试的门槛相较于手工测试较高，它要求自动化测试开发人员具有对测试环境进行搭建、并编写自动化测试脚本的编程能力。

所以，对自动化测试的研究以及如何应用在企业项目开发中的软件过程管理中的探讨势在必行，而使用SaaS软件服务提供BDD风格的自动化测试管理平台，将为自动化测试提供一个标准、严格、合乎软件过程管理规范的管理与执行工具。

1.2国内外研究现状

据Oppenheimer Funds 的调查，在2001年前后三年中，全球范围内由于采用了测试自动化手段得到的ROI（投资回报率）高达1500%。随着对测试的要求越来越高以及软件测试技术的发展，实现软件测试自动化的趋势已经不可逆转了。

国外对敏捷过程管理中测试驱动开发以及能够理解用普通语言描述的测试用例的支持BDD的Cucumber的理论研究、实践，侧面证明了自动化测试对于软件开发的质量管理的重要性。而在国内自动化测试属于较为新鲜的技术，在研究方面稍为落后，并且缺少丰富的应用经验。所以自动化测试在国内许多公司处于一个尴尬的境地：大部分公司依旧重度依赖手工测试，或想开展自动化测试却发现并没有收获预想中的效果。

在企业实施自动化过程中，绝大部分中国企业用户选择在企业内部实现测试自动化，选择在企业中开展自己的自动化测试服务，并且成立自己的自动化测试部门来组建自己的自动化测试系统工作。而在国外，由于国外SaaS软件的盛行，国外的企业（尤其是中小型企业）往往是通过选择由专门企业提供的测试自动化平台托管服务，并对此服务付费，或是通过外包、派遣、离岸等多种方式相结合来实现对自动化测试的需求。国外的企业关注测试带来的结果、发现的问题，而不是由如TestingWhiz等专业的提供测试服务的公司提供的测试的技术、过程。这相当一部分原因是因为国内外的软件技术发展趋势的不同与SaaS软件并没有在国内得到很好的发展。

1.3 研究内容和论文结构

（1）本文致力于对

第二章 软件测试概论

2.1 软件测试的概念与目的

软件测试是在有限的时间中保证软件高质量的重要手段，是软件开发过程中相当重要的一部分。测试人员通过特定的手段，针对软件的功能或模块，按照一定的流程来对软件进行执行并找出其中的错误，这是软件测试的基本方式。

IEEE中给出了软件测试的定义：软件测试是使用人工或自动手段来测试某个软件的过程，它的目的在于检查软件是否符合规定并满足需求，或者是对比并找出被测软件的预期行为与实际行为之间的差别。

可以看出，软件测试的目的主要是衡量系统是否符合预期并满足用户的需求。换言之，如果软件系统与用户需求不符，则可判定其中存在缺陷，因为用户需求并不能以系统意志所转移，所以系统将需要得到修正。但是，软件测试的目的并非是追求完美切100%无错的系统，在测试人力、资源上的投入需要满足供需关系，当系统已经趋近无错时，发现漏洞与问题已经愈发困难；而投入过多精力在测试上，则有可能导致不必要的成本增长。

软件测试的通常方式是设计测试用例，并比较实际行为或输出结果与预期行为或预期输出结果之间的差异，且此差异是否在用户可以接受的范围之内。在测试执行的阶段，需要注重测试过程中的缺陷，对用例的执行情况进行评估，以求为发现系统的边界值等极端错误进行合理的测试，来设计优秀的测试用例。

2.2自动化测试

自动化测试是软件测试中的一个有别于手工测试存在的概念，通过相应的测试工具或测试脚本来运行大量测试用例以完成测试计划。它能够完成许多手工测试难以做到的测试方法，并且能够自动地生成测试报告与结果，提示错误信息，从而节省了大量的人力资源与时间。顾名思义，自动化测试是通过机器取代人工，以拟定的方式运行程序，并对程序的结果进行比对以达到测试的目的，从而提高软件质量与可靠性的测试方法。

自动化测试相对于手工测试有许多的优点，使得自动化测试在现代软件开发中拥有至关重要的地位：

1. 自动化测试可以减少重复性的测试工作，节约测试成本。在项目的版本迭代中，或是在项目新功能的添加中，由于代码存在着一定的耦合性，代码之间有可能在一定程度上存在着互相的影响。例如在开发中，由于数据库设计时没有全面的考虑，导致在新功能增填时需要修改数据库的字段，从而导致旧代码可能存在着一些问题。所以，在版本的迭代中，测试人员常常需要对系统进行多次的回归测试（修改了旧代码后，对系统进行重新测试以确认此修改没有引入问题或带来其他的影响），而在大型系统中，完整的软件生命周期中回归测试更是不可少的一部分，在软件开发的各个阶段，都需要多次的回归测试。在整个软件测试的过程中回归测试占了极大的比重，而回归测试在一定程度上可以说是大量的重复性的劳动，它给手工测试带来了许多困难，极大地增加了手工测试的成本与时间的浪费，尤其是在渐进开发与开速迭代开发中，回归测试更加频繁，在极限编程中，一天甚至有可能要进行好几次回归测试。这使得手工测试的缺陷被放大，而自动化测试则能很好地适用于此场景。由于自动化测试脚本的可维护性、可重用性，测试开发人员不需要重新开发测试用例对应的测试脚本，只需要让测试脚本再次运行即可。这样节约了测试成本与时间，缩短项目开发周期。
2. 在相当一部分情况下，自动化测试能够完成许多手工测试无法完成的测试。例如在对网站、App的负载均衡测试中，人工无法模拟大量的（例如百万级别）访问、使用数据，而在自动化测试中可以通过计算机的执行来模拟出大量的访问，可以对软件进行性能测试或者说压力测试。许多测试工具如LoadRunner即是在程序内部封装了访问的代码，并通过配置决定需要的访问量，来做对软件的性能测试与性能分析。而在多线程并发中，bug往往难以复现，而手工地测试更是基本不可能找出错误，只有自动化测试可以模拟微秒级别的并发问题。所以，自动化测试是对高性能需求项目开发中不可缺少的一部分。这也是自动化测试开发的主要意义所在。
3. 自动化测试能够提高软件质量的可信度。手工测试在人力完成的情况下不足以对软件质量进行十分严谨的保证，而自动化测试在单元测试中，可以提供高覆盖率、高精度的保证。在自动化测试用例下，由于脚本由机器执行，所以减少了执行过程中的错误与疏漏问题，而将问题关注到测试用例的设计质量中。特别当在复杂的计算中，自动化测试具有手工测试无法比拟的优势。
4. 自动化测试具有很高的复用性、一致性、可重复性。自动化测试用例往往采用脚本技术，而在较多的不同环境下，所需要编写的测试脚本都是类似的，这使得测试脚本具有很好的可复用性，从而使得编写自动化测试脚本所需时间成本降低。而脚本的执行是由计算机来完成，脚本的输入一致的情况下，输出在没有随机的情况下也是一致的，这很好地保证了测试用例的一致性。而脚本代码往往可以保存下来再次使用，可以提供高频率地重复运行来验证软件的可靠性，同时节省了重复执行同一个测试用例的时间，将测试人员的时间集中在新的特征功能中。

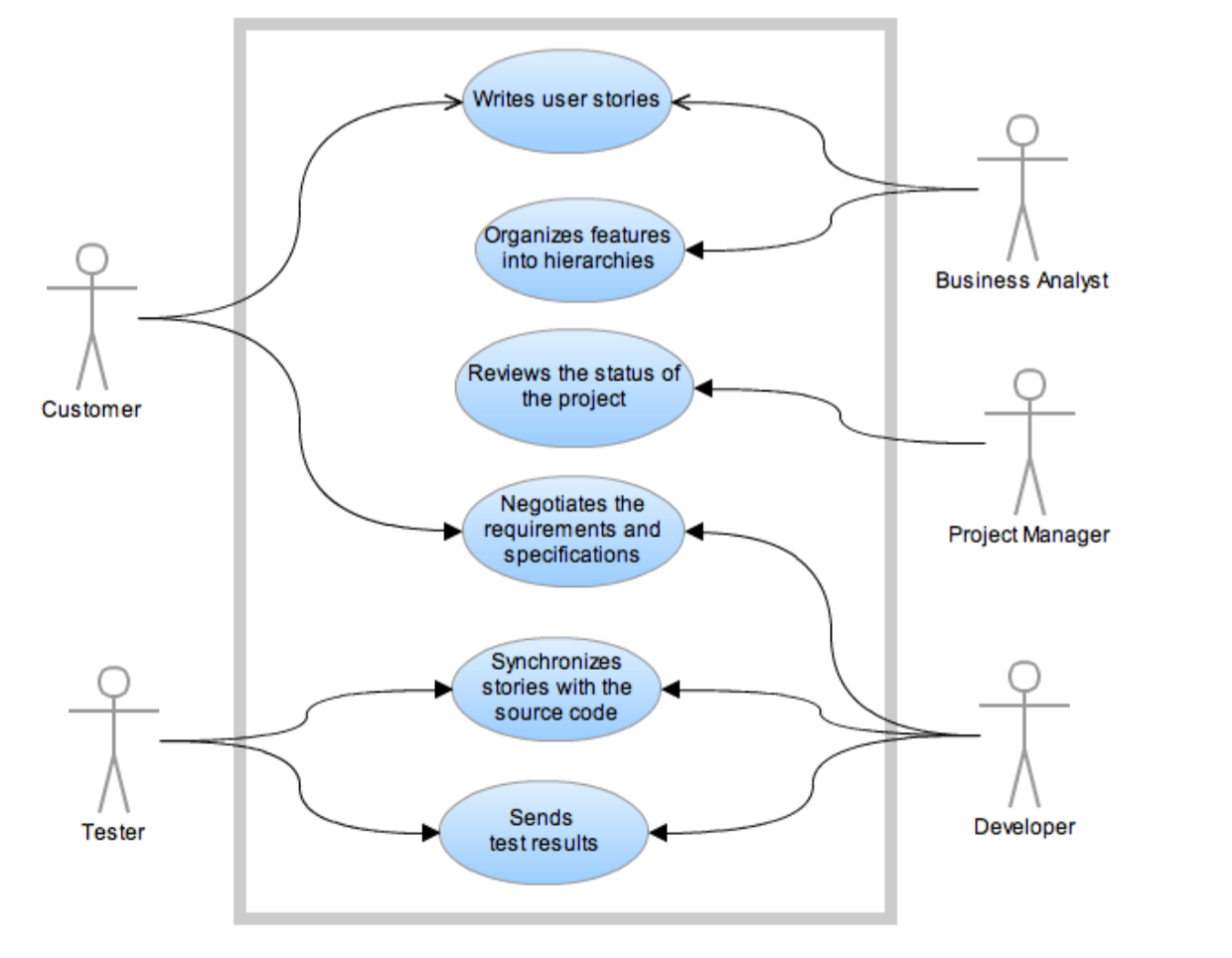
然而自动化测试仍然不能完全取代手工测试。在许多主观性的场景，手工测试还是必不可少的。例如判断web页面中Css样式是否正常、合理，是自动化测试难以完成的。而且手工测试人员在测试过程中可以发现许多测试用例之外的错误或问题，以编写更加完善、有意义的测试用例，这对测试用例为核心的测试过程是十分有意义的。而且，编写测试用例需要测试开发人员具有较好的编程水平，而在某些复杂场景中编写自动化测试脚本往往会花费许多时间，而是否使用自动化测试需要妥善的衡量利弊。

综上所述，自动化测试在提高项目质量与可靠性中具有许多优点，但是自动化测试只是手工测试的补充，它不能取代手工测试，在项目中如何使用自动化测试和手工测试，需要依据具体情况来定义。

2.3BDD与软件测试

BDD（行为驱动开发）是在TDD（测试驱动开发）理念基础上发展的一种敏捷开发方式，它鼓励开发工程师、测试工程师与客户（需求提供者）之间的协作，包括验收测试与客户测试驱动等极限编程的实践，作为对测试驱动开发的回应。2009年，Dan North对行为驱动开发给出了定义：行为驱动开发是由外而内的、需要多方协同合作的、可拓展的、高自动化的第二代敏捷方法。2010年，Chelimsky将BDD具体描述为：行为驱动开发是通过从利益相关者（如客户）的角度描述其行为来实现应用程序的开发方法。

BDD的开发一定程度上借鉴了TDD的思想。TDD是指在编写功能性代码之前，首先选择先编写测试用例，根据测试的需求推动开发的需求。它很好地诠释了“代码即文档”的理念。但是在TDD开发中，由于测试代码往往是程序代码的两倍，并且在开发人员在进行单元测试时，往往是对代码的结构单元，如类或方法进行测试，然而在编写测试时，开发人员往往对代码结构只有一个模糊的想法。此外，测试以来代码结构，使得代码难以重构。

2005年，Astels批评了TDD的不足，并提出了使用行为与规范代替测试，关注软件做什么而不是验证软件是否正确-----即行为驱动开发。行为驱动开发方法提供了一种描述预期行为的方式------使用通用的自然语言描述系统行为，使得项目开发人员与项目的测试人员、需求提供者与产品经理对项目有一致的了解，并允许QA团队从利益相关者的角度关注系统。BDD强烈地关注了客户与客户需求，力求开发出的产品在最大程度上与需求提供者描述的预期相符，从而提高项目质量。 然而，