**基于Selenium的Web自动化测试**

**1 绪论**

**1.1引言**

网络时代的到来和迅速普及，为软件产业带来了一场革命性的变化，基于Web的应用系统已经开始逐步取代原来的单机版应用系统，成为当前和未来的软件系统开发和实施的主流。现在的Web应用系统结合了商业、数据库以及企业的运用，因此，对于Web应用系统的要求也愈来愈严格，它必须具备高度的扩展性，合理的执行效率，以及全天候安全强固的执行环境。也就是说，现在Web应用系统必须能够安全及时地服务大量的客户端用户，又能够长时间安全稳定地运行。而且由于interent的开放性和易访问性，在Web应用系统商业应用领域的竞争非常激烈。用户对网站的期望很高，如果网站无法做到快速加载、正确显示信息、即时反应并提供直观的浏览与简易的交互功能，用户就有可能转换门庭，去别的网站。因此，Web应用的测试至关重要。

但是由于Web应用系统具有分布、异构、并发和平台无关的特性，因而Web应用系统的测试要比普通程序的测试要复杂的多。

从功能测试角度看，与传统的应用软件相比，Web应用系统的独特之处主要有以下几点：

1．Web应用系统的组成实体多种多样。就HTML文档而言，用不同语言编写的脚本，各式各样的样式表及组件使得Web应用系统难以理解和测试。

2．Web应用系统中有大量的导航链接，确保系统能够根据用户的选择准确地显示用户需要的内容是Web测试的重要方面。

3．Web应用系统通过会有大量的Cookie等技术来保存用户的状态信息，确保系统对这些状态信息的正确管理也是Web测试的重要挑战。

4．Web应用系统的客户端及操作系统的多样性导致的兼容性问题，要求对各个环境进行测试。

5．Web应用系统客户端内容及结构更新快，新功能的不断加入不仅要对新加入功能进行测试，而且还要对原有功能进行回归测试。

6．Web应用系统开发技术层出不穷，如AJAX，XML，JSP/ASP等等，每种技术都有各自的特点，测试也要根据相应的技术进行以提高测试效率。

 相对于传统软件而言，基于Web应用系统的测试工作要求相对来说更高，测试工作更严峻，如果单靠手工去进行测试，是远远不够的，必须引入自动化测试方法。

**1.2 Web应用系统功能测试自动化研究**

在实际测试工作中，自动化测试的难点在于：

1．测试用例。测试用例是手工测试与自动化测试的基本，自动化测试效率的高低与测试用例的设计的好坏有着直接的关系。

2．自动化测试用例的选取。哪些测试用例适合用来进行自动化呢？只有选取那些重复性高，手工测试无法实现或自动化测试比手工测试效率更高的测试用例进行自动化，才能更好的提高自动化测试的效率。但是如何确定测试用例符合这些标准又是一个值得讨论的问题。

3．自动化测试脚本的健壮性。Web应用系统的表现层的内容与结构的易变性经常会导致测试脚本失败，而后又需要维护，这样大大降低了自动化测试的效率。

 目前，对于Web应用系统的功能测试基本上都是采用“录制一回放”的模式来生成测试脚本，即测试人员用自动化测试工具将手工执行的测试过程录制成测试脚本，然后根据需要修改测试脚本，接着执行测试脚本，如果测试脚本执行失败，分析失败的原因，如果是测试脚本的问题，就需要对测试脚本进行维护。

对于元素的定位地址，目前很多的自动化测试工具一般或使用DOM，或使用XPath表达式来表达页面元素的定位信息，如：//ul[@，class＝'in']/LI[1]/a，可读性差，测试人员需要到页面中去定位该元素才能了解该元素代表什么。而且元素的定位信息遍布于测试脚本中，一旦该元素的定位信息有所改变，就需要首先找出使用到该元素的测试脚本，并逐一更新定位信息。

 我们需要对页面元素的定位信息进行统一管理，使用一个有意义的名称去标识被测页面元素，对每一个页面元素的定位信息应只有一个单一的入口点，并通去一定的定位逻辑动态的生成页面元素的定位信息。

**1.3研究内容和组织结构**

本文针对Web应用系统的特点，首先介绍了软件测试的方法及Web应用系 统测试方法，自动化测试及其应用，根据自动化测试工作中出现的页面元素定位难的问题，基于开源自动化测试工具Selenium，提出了一套解决方案selenium+junit框架，以解决web自动化测试的问题。

本文的组织结构如下：

第一章，简述了Web应用系统自动化测试的背景知识及其现状，并介绍本文的研究内容。

第二章，首先介绍了软件测试的基本概念，并分析了Web应用程序的特点，及Web应用程序测试的主要方法。

第三章，首先介绍了软件测试自动化的基本概念，并就Web应用程序的特点分析了自动化测试与手工测试的不同。然后介绍自动化测试工具的选择及应用。

第四章，首先介绍了开源功能测试工具Selenium及如何有效使用该工具，然后构建Selenium + Junit框架进行自动化测试。

第五章，对本论文的研究进行总结，讨论了它的可取与不足之处，指出进一步研究的方向。

**2 软件测试概述**

**2.1 什么是软件测试**

**2.1.1 软件测试的定义**

 所谓软件测试，就是一个过程或一系列过程，用来确认计算机代码完成了其

应该完成的功能，不执行其不该有的操作。软件应当是可预测且稳定的，不会给

用户带来意外惊奇。

 软件测试就是在软件投入运行前，对软件需求分析、设计规格说明和编码的

文档和程序最终复审，是软件质量保证的关键步骤。如果给软件测试下定义，可

以这样讲：测试是为了发现错误而执行程序的过程，也就是说，软件测试是根据

软件开发各阶段的规格说明和程序的内部结构而精心设计的一批测试用例（即输

入数据或操作及其预期的输出结果），并利用这些测试用例去运行程序，以发现

程序错误的过程。

软件测试是为了发现程序中的错误，是一个找错的过程；软件测试的过程亦

是程序运行的过程；程序运行需要数据，为测试而设计的数据或辅助程序称为测

试用例。

 值得指出的是，不能保证通过测试的程序一定正确，测试只能找出程序中的

错误，而不能证明程序无错。有时可以认为，软件运行期间测试活动从未间断，

只是在软件交付之后，将由用户来继续扮演测试角色而已。

**2.1.2 软件测试的目标**

对于软件测试，基于不同的立场，存在着两种完全不同的测试目标。从用户

的角度出发，普遍希望通过软件测试暴露软件中隐藏的错误和缺陷，以考虑是否

可以接受该产品；而从软件开发人员的角度出发，则希望测试成为表明软件产品

中不存在错误的过程，验证该软件产品已正确地实现了用户的要求，确立人们对

软件质量的信心；因此，他们会选择那些导致程序失效概率小的测试用例，回避

那些易于暴露程序错误的测试用例；同时，也不会着意去检查、排除程序中可能

有的副作用。显然，这样的测试对完善和提高软件质量是毫无价值的。由于，程

序中往往存在着许多预料不到的问题，可能会被疏漏，许多隐藏的错误只有在特

定的环境下才有可能暴露出来；如果不把着眼点放在尽可能查找错误这样一个基

础上，这些隐藏的错误和缺陷就查不出来，会遗留到程序运行阶段中去。如果从

用户的角度考虑，替他们着想，就应当把测试活动的目标对准揭露程序中存在的

错误；在选取测试用例时，需要考虑那些深层次意义下才可能发现程序错误的数

据。

有些测试专家认为软件测试的范围应当包括得更广泛些。认为软件测试不仅要考虑正确性以，还应当关心程序的效率、健壮性等因素，并且应该为程序调试提供更多的信息。20世纪70年代中期以来，形成了软件生命期概念。这时人们对于软件测试的认识更广泛，也更深刻了，这对于软件产品的质量保障以及组织好软件开发工具有着重要的意义。这时，对软件质量的判断决不只限于程序本身，而是整个研制过程。

**2.1.3软件测试的原则**

测试是一项非常复杂的、创造性的和需要高度智慧的挑战性的工作。测试一

个大型程序所要求的创造力，可能要超过设计那个程序所要求的创造力。软件测

试中很重要的一个方面是人的心理问题，一些直观上看是很显而易见的至关重要

的原则，总是被人们忽视。

确定预期输出（或结果）是测试情况必不可少的一部分，如果事先无法肯定预期的测试结果，往往会把看起来似是而非的东西当成正确的结果。必须提倡用事先精确对应的输入和输出结果来详细检查所有的输出。

程序设计机构不应测试自己的程序，程序员也应避免测试自己的程序。软件

测试的出发点是找错误，要让程序设计机构和程序员在测试自己的程序时持否定

的态度是困难的。除了这个心理学问题外，还要注意：程序中可能包含由于程序

员对问题的叙述或说明的误解而产生的错误。如果是这种情况，让程序员测试自

己的程序时往往是不能发现问题的。以上的看法并不意味着程序设计机构或程序

员不能调试自己的程序，而是强调，由第三方来进行程序测试会更有效、更成功。

一般而言，测试计划可以在需求分析完成后开始，详细的测试用例定义

可以在设计模型被确定后立即开始，因此，所有测试可以在任何代码被

编写前进行计划和设计。

>Pareto原则应用于软件测试。Pareto原则意味着测试发现的错误中的80％

很可能集中在20％的程序模块中。

>测试应从“小规模＂开始，逐步转向“大规模”。即从模块测试开始再进

行系统测试。

>穷举测试是不可能的，因此，在测试中不可能覆盖路径的每一种组合，

然而，充分覆盖程序逻辑，确保覆盖程序设计中使用的所有条件是有可能的。

>为达到最佳的测试效果，提倡由第三方来执行测试。

**2.1.4软件测试的对象**

软件测试并不等于程序测试。软件测试应当贯串于软件定义与开发的整个期

间；因此，需求分析、概要设计、详细设计以及程序编码等各阶段所得到的文档，

诸如：需求规格说明、概要设计规格说明、详细设计规格说明和原程序等，都应

成为软件测试的对象。软件测试不应仅局限在程序测试点的狭小范围内，而置其

它阶段的工作于不顾。

**2.2软件测试的基本方法**

软件测试方法可根据测试过程中是否发生状态变化分为两大类：动态测试和

静态测试方法；又可根据对测试对象了解的程度，分为黑盒测试和白盒测试两类。

**2.2.1静态和动态的方法**

根据程序是否运行，测试可分为静态测试和动态测试。静态测试就是静态分

析，对模块的源代码进行研读，查找错误或收集一些度量数据，并不需要对代码

进行编译和仿真运行。静态测试采用人工检测和计算机辅助静态分析手段进行检

测，只进行特性分析。

静态测试包括对软件产品的设计规格说明书的审查，对程序代码的阅读、审

查等。静态分析的查错和分析功能是其它方法所不能替代的，已被当做一种自动

化的代码校验方法。

动态测试是通过观察代码运行的动作，来提供执行跟踪、时间分析，以及测

试覆盖度方面的信息。动态测试通过真正运行程序发现错误。通过有效的测试用

例，对应的输入/输出关系来分析被测程序的运行情况。

不同的测试方法各自的目标和侧重点不一样，在实际工作中，应将这两种方

法结合起来运用，以达到更完美的效果。

**2.2.2黑盒测试和白盒测试**

根据测试是针对系统的内部结构还是针对具体实现算法的角度来进行，分别

称为白盒测试和黑盒测试。

1．黑盒测试法

黑盒测试，也称功能测试或数据驱动测试。它不管程序内部结构是什么样的，只是从用户出发，根据产品应该实现的实际功能和已经定义好的产品规格，来验证产品应该具有的功能是否实现，每个功能是否都能正常使用，是否满足用户的要求。

在测试时，把程序看作一个不能打开的黑盒子，在完全不考虑程序内部结构 和内部特性的情况下，测试人员针对程序接口和用户界面进行测试，只检查程序

功能是否按照需求规格说明书的规定正常使用，程序是否能适当地接收输入数据

而产生正确的输出信息，并且保持外部信息（如数据库或文件）的完整性。

黑盒测试主要用于发现以下情况：

>是否有不正确或遗漏了的功能；

>在接口上，能否正确地接受输入数据，能否产生正确的输出信息；

>访问外部信息是否满足要求；

>性能上是否满足要求；

>界面是否错误，是否不美观；

>初始化和终止错误。

黑盒测试方法主要用于软件确认测试。其具体方法有等价类划分、边界值分

析、因果图、错误猜测法等。

黑盒测试方法着眼于程序外部结构，不考虑内部逻辑结构，针对软件界面和软件功能进行测试。在用黑盒测试时，必须在所有可能的输入条件和输出条件中确定测试数据。黑盒测试中不可能做到穷举测试，因此局限于功能测试是远远不够的，还要结合白盒测试方法，进行逻辑和路径测试。

2．白盒测试法

白盒测试，也称结构测试或逻辑驱动测试，也就是已知产品的内部工作过程，

清楚最终生成软件产品的计算机程序的结构和语句，按照程序内部的结构测试程

序，测试程序内部的变量状态、逻辑结构、运行路径等，检验程序中的每条通路

是否都能按预定要求正确工作，检查程序内部动作或运行是否符合设计规格要

求，所有内部成分是否按规定正常进行。主要用于软件验证。白盒测试的主要方

法有逻辑覆盖、循环覆盖和基本路径测试。逻辑覆盖又可分为语句覆盖、判定覆

盖、条件覆盖、判定/条件覆盖、条件组合覆盖和路径覆盖。

白盒法是“基于覆盖的测试＂，应朝着提高覆盖率的方向努力，尽可能多地

进行测试，找出那些被忽视的错误。一般来说，白盒测试的原则是：

>保证每个模块中所有独立路径至少被使用一次。

>对所有逻辑值均测试为真值和假值。

>在上下边界及可操作范围内运行所有循环。

>检查内部数据结构以确保其有效性。

**2.3软件测试的阶段**

软件测试是软件开发过程中的重要内容之一，是软件质量保证的关键。软件

测试贯穿软件产品开发的整个生命周期一一软件项目一开始，软件测试也就开始

了，从产品的需求分析审查到最后的验收测试、安装测试结束。整个过程如图

2-1所示。

设计阶段

编码及单元白盒测试

阶段

单元黑盒测试阶段

集成测试阶段

确认测试阶段

系统测试阶段

安装测试阶段

验收测试阶段

结束总结阶段

**图2－1 软件测试阶段示意图**

从过程来看，软件测试是由一系列的不同测试阶段所组成的，这些阶段分为：

单元测试、集成测试（组装测试）、系统测试和验收测试。软件开发的过程是自顶向下的，测试则正好相反，以上这些过程就是自底向上，逐步集成的。当然，这里只是一个完整的过程，对于不同的软件系统或产品可以进行适当的裁减或合

并。

**2.4软件测试的工作范畴**

软件测试工作可以分为六个方面：

测试组织和管理：建立测试队伍，设立不同功能或完成不同任务的测试小组，

对测试用例、软件缺陷、测试执行、测试文档等进行管理，当然，也可以把测试

管理工作看成是软件质量管理工作的一部分。

测试计划：独立的测试组织负责定义软件测试的方法和规范。开发组织负责

编制单元测试的计划和说明。测试组织主要负责编制其它各测试阶段的测试计划

和说明。

设计测试用例：为了更有效地进行测试，需要设计测试用例。

测试实施：按测试计划与测试说明的定义对测试对象进行相应的测试，填写

测试报告中相应的表格，并通知相关人。

测试结果分析：对测试结果进行定量和定性的分析，以检查测试工作执行的

状态。

测试评审与报告：依据软件测试评审准则在各测试阶段评审时提交类型完整

的测试文档。

**2.5Web应用系统的特点及测试内容**

 **2.5.1Web应用系统的特点**

Web应用系统基本是建立在客户/服务系统之上的，在这种系统中，客户端是各种可以访问因特网的终端设备，如手机，个人计算机，掌上电脑等等，

但绝大多数对应用系统的访问都是来自个人计算机上的浏览器。

由于各个浏览器对颁布的标准的遵循度，以及对标准的扩展方面大相径庭，导致同一个页面在不同的浏览器上的显示及逻辑处理上都不相同，开发人员在开发基于Web应用程序时就需要考虑到不同浏览器之间的不同，及对不同的脚本处理上的不同，浏览器兼容性的测试也是Web应用系统测试中的一个重要挑战。

Web应用系统的服务端一般有三层：

第一层：表示层。用户对Web应用系统的直观体验就在一层上，在这一层

上将提供各种用户需要的信息及交互操作，并把用户的请求发送到应用服务端，

并把相应的结果返回给用户。

第二层：业务层，运行应用服务器。在这里运行的软件模拟业务流程。下面

列举的是一些与业务层有关的功能：事务处理；用户身份鉴定；数据确认；程序

日志，大型的Web应用系统在此层会加入负载均衡机（Load Balance）分散服务器的压力。

第三层：数据层，在这里从一个或多个关系数据库管理系统（RDBMS）中存储和获取数据，它包含与第二层进行通信的数据库设备。进入数据层的接口由数据模型定义，模型描述了如何进行数据存储。该层一般也会用到负载均衡机制以缓解应用系统的高事务量。

**2.5.2Web应用系统的测试内容**

Web测试主要通过功能测试、性能测试、兼容性测试、安全性测试与可用性

测试等来发现Web系统的缺陷。

1．功能测试

功能测试就是结合规格说明的要求，保证功能上正确无误。实现用户功能需

求是对软件系统最基本的要求，所以软件功能测试也是所有测试任务的基础。抛

开与传统软件系统功能测试的相同之处，Web应用系统功能测试主要包括以下五

方面的内容：链接测试、表单测试、COOKIE测试、脚本语言测试、数据库测试。

在制定测试计划、设计测试策略、选择测试方法和分配人力资源等方面，功能测

试往往是考虑最多的因素。

2．性能测试

通过测试确定系统运行时的性能表现，如得到运行速度、响应时间、占有系

统资源等方面的系统数据。如某个网站可以被访问，而且可以提供用户预先设定

的功能，意即通过了功能测试，但第打开一个页面或完成某个请求都需要1-2

分钟，用户不可忍受，其结果没有用户愿意使用这个网站所提供的服务。

3．配置和兼容性测试

因为Web应用系统通常需要支持多个主流的Web浏览器，而最终用户面对

的也是这些不同的浏览器。针对浏览器的配置和兼容性测试在Web应用系统测

试中必不可少，并且占了Web应用系统客户端配置和兼容性测试的大部分时间。

所以设计有效的测试用例，精简宠大的测试矩阵是Web兼容性测试的要点。

4．可用性测试

满足用户需求，易于用户使用的Web应用系统才是好的系统。Web应用系

统最终要面向Web用户，系统给用户的印象是非常重要的，如果网站不规范、

很难使用或者功能不能运行，将会使用户对网站失去兴趣，转去访问别的网站，

最终失去用户。

由此可见，Web应用系统的测试工作量更大，仅仅靠手工测试去完成，效率

低，而且测试人员资源宝贵应该投入更多需要智力的工作上，如测试用例的设计

等，自动化测试可以完成那些机械的、重复的测试工作或手工无法完成的测试工

作，如性能测试。引入自动化测试可以大大提高测试工作的效率。

**3 基于Web应用程序的自动化测试**

**3.1什么是软件自动化测试**

所谓自动化测试，就是执行自动测试工具或者用某种程序设计语言编制的自

动测试程序，控制被测软件中的各种类和对象，模拟手动测试步骤，完成测试工

作。自动化测试的根本目的是自动地对软件产品在各种环境和状态下的执行进行

测试，排除影响测试的人为因素，从而降低花费在测试上的开销。

自动化测试方法的优点包括可以大规模地提高测试效率，减少测试工作量；

具有可重复性，可精确地再现以前的测试步骤，有利于进行回归测试；可以降低

人为的操作失误，从而减少测试成本。

传统的手工测试是一个劳动密集型工作，可以充分利用人的能力。测试员可

以临时想出新测试，也可以注意到没有或不能预测的现象。但是手工测试容易出

错。它不支持那些可以由自动测试完成的相同种类的质量检查，引入自动测试能

够用更有效、可重复的自动测试环境代替手工测试活动。

虽然自动测试能够增强测试工作的效率，但是不能利用测试员隐含的知识和

认识。自动化测试每次运行都以同样的速度、同样的顺序、完全一样的鼠标移动

和键盘操作去做同样的事。而手工测试在每次运行测试时都可以对测试做变动。

这些变动可以发现未看到的程序错误。因此，一味在项目测试中追求全部测试自

动化是不现实的也是不合理的，应该把自动化测试看作是手工测试的扩充，用

其来完成手工测试所不能完成的工作。

**3.2软件自动化测试的背景及意义**

**3.2.1测试自动化的背景**

最近行业内对软件的质量要求越来高， 这必然引起了对测试工作的重视，一款好软件的出世，不但要求有强大的测试团队，该团队必须具备在业务方面、测试技能方面的专业水平， 而且在软件开发过程方面经常由于测试而作持续不断地调整。

幸运的是，随着软件开发技术和工具的提高，软件工程和软件过程实践的推广， 软件测试日益得到重视和专业化。 我别论文期间，一直研究测试理论、自动化测试工具，并建立了一套测试体系。

在此并不介绍整个测试体系，而是介绍测试方面最值得探讨的部分：测试自动化。目的是与大家共同进步。当然自动化测试方面的介绍,但我要介绍的是自动化测试,而是测试如何的自动化.

**3.2.2自动化测试实施的意义**

1、 对新版本执行回归测试－测试每个特征

对于产品型的软件，每发布一个新的版本，其中大部分功能和界面都和上一个版本相似或完全相同，这部分功能特别适合于自动化测试， 从而可以让测试达到测试每个特征的目的。

2、 更多更频繁的测试－沉闷、耗时

我们的产品向市场的发布周期是3个月，也就是我们的开发周期只有短短的3个月，而在测试期间是每天/每2天都要发布一个版本供测试人员测试，一个系统的功能点有几千个上万个，人工测试是非常的耗时和繁琐，这样必然会使测试效率低下。

3、替代手工测试的困难

压力测试、并发测试、大数据量测试、崩溃性测试，用人来测试是不可能达到的。

4、具有一致性和可重复性

由于每次自动化测试运行的脚本是相同的, 所以每次执行的测试具有一致性, 人是很难做到的. 由于自动化测试的一致性，很容易发现被测软件的任何改变。

5、更好的利用资源

理想的自动化测试能够按计划完全自动的运行, 在开发人员和测试人员不可能实行三班倒的情况下, 自动化测试可以胜任这个任务, 完全可以在周末和晚上执行测试. 这样充分的利用了公司的资源,也避免了开发和测试之间的等待.

6、解决测试与开发之间的矛盾

常在开发的末期,进入集成测试阶段, 由于每发布一个版本的初期,测试系统的错误比较少,这时开发人员有等待测试人员测试出错误的时间. 事实上在叠代周期很短的开发模式中，存在更多的矛盾， 但自动化测试可以解决其中的主要矛盾。

7、增加软件信任度。

**3.3软件自动化测试的优点**

软件自动化测试一般通过执行某种程序设计语言编制的自动测试程序，控制

被测软件的执行来模拟手动测试步骤，完成全自动或半自动测试，其目的在于

缩短测试周期，增强对软件性能方面的测试能力等，其优点可总结为以下几点：

（1）令软件新版本进行回归测试的开销最小。

（2）可以在更短的时间内完成更多的测试。

（3）可以完成一些手工测试不能或难以完成的测试。

（4）具有一致性和可重复性。

（5）更好地利用资源。

**3.4自动化测试的引入条件**

自动化测试能大大降低手工测试工作，但决不能完全取代手工测试。完全的

自动化测试只是一个理论上的目标，实际上想要达到100％的自动化测试，目前不可能实现。正常40％的自动化测试度已经是非常最高的了。

测试自动化的引入有一定的标准，要经过综合的评估，绝对不能理解成测试

工具简单的录制与回放过程。实际上，从实现成熟度来说，自动化测试可分为以

下五个级别（表3－1）：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 级别 | 说明 | 优点 | 缺点 | 用法 |
| 一级 | 录制和回放 | 自动化的[测试脚本](http://www.ltesting.net/html/96/category-catid-96.html)能够被自动的生成，而不需要有任何的编程知识 | 拥有大量的测试脚本，当需求和应用发生变化时相应的测试脚本也必须被重新录制 | 当测试的系统不会发生变化时，实现小规模的自动化 |
| 二级 | 录制、编辑和回放 | 减少脚本的数量和维护的工作 | 需要一定的编程知识；频繁的变化难于维护 | [回归测试](http://www.ltesting.net/html/72/category-catid-472.html)时，用于被测试的应用有很小的变化 |
| 三级 | 编程和回放 | 确定了测试脚本的设计，在项目的早期就可以开始自动化的测试 | 要求测试人员具有很好的软件技能，包括设计、开发 | 大规模的测试套件被开发、执行和维护的专业自动化测试 |
| 四级 | 数据驱动的测试 | 能够维护和使用良好的并且有效的模拟真实生活中数据的测试数据 | 软件开发的技能是基础，并且需要访问相关的测试数据 | 大规模的测试套件被开发、执行和维护的专业自动化测试 |
| 五级 | 使用动作词的测试自动化 | [测试用例](http://www.ltesting.net/html/94/category-catid-94.html)的设计被从测试工具中分离了出来 | 需要一个具有工具技能和开发技能的测试团队 | 专业的测试自动化将技能的使用最优化的结合起来 |

**表3－1 自动化测试的引入级别**

自动化测试能提高测试效率，快速定位测试软件各版本中的功能与性能缺

陷，但不会创造性的发现测试脚本里没有设计的缺陷。测试工具不是人脑，要求

测试设计者将测试中各种分支路径的校验点进行定制；没有定制完整，即便事实

上出错的地方，测试工具也不会发觉。因此，制订全面、系统的测试设计工作是

相当重要的。

自动化测试能提高测试效率，但对于周期短、时间紧迫的项目不宜采用自动

化测试。推行自动化测试的前期工作相当庞大，将自动化测试框架应用到一个项

目中也要评估其合适性，因此决不能盲目的的应用到任何一个测试项目中，尤其

不适合周期短的项目，因为很可能需要大量的测试框架的准备和实施而会被拖跨。

**3.5自动化测试的可行性分析**

任何自动化测试在项目中的应用都要结合项目本身的特点和测试的实际需

要。对于系统功能测试，项目的需求分析和设计分析文档以及测试用例都是进行

自动测试可行性分析的基础。在这些因素确定以后，接下来需要从以下几方面

进行引入自动化测试的可行性分析。

1．对测试用例进行分类

自动化测试要以手工测试的测试用例为基础，因此把整个测试用例按照自动

化测试工具实现程度的难易分成以下四个不同的等级：

（1）不能实现的测试用例。

（2）实现困难的测试用例。

（3）实现一般的测试用例。

（4）实现容易的测试用例。

2．比较自动化测试和手工测试的工作量

使用自动化的测试的主要目的之一就是降低手工测试的工作量，因此如果引

入自动化测试后引起的工作量的增加远大于原来手工测试的工作量，就是不经济

的，这种情况下我们就不应考虑引入自动化测试。因此，在一开始的准备阶段，

对于那些可以由自动测试完成的测试用例，需要分别进行手工测试和自动测试的工作量记录和比较、分析，得出自动测试对于项目的可行性程度。由于该部分内容和本文主题没有直接关系，因此不再展开进行论述。

基于以上两方面分析和实际的使用经验，可以得到影响自动测试技术是否适

合项目的可行性分析的因素：

>项目组测试用例的循环测试轮数，一般只要两轮以上的测试，自动化测

试的消耗将小于手工测试的消耗，同时越多轮数的测试就越适合使用自动测试技术，自动测试技术的可行性程度越高；

>测试组对测试用例的分析需要结合测试工具的实际功能，越容易用测试

工具实现测试的用例越多，自动测试技术的可行性程度越高

>自动测试效率受到测试人员对测试用例的熟悉程度，测试人员对于自动

测试工具熟练使用程度以及测试人员与开发人员的沟通能力等因素，这些因素都将影响到自动测试的可行性程度。

**3.6软件自动化测试在软件行业的使用现状**

自动化软件测试研究已经逐步进入成熟阶段，各种商业自动化测试工具也相

继被推出，有越来越多的软件开发组织开始尝试在测试过程中引入自动化测试。

但是具体的实施情况参差不齐，按实施的不同层次可以分为以下几类：

测试自动化没有纳入规划。认为不必实施自动化测试，主要是软件开发组织

的人员、资金、资源都不足。

准备实施测试自动化的过程中遇到阻力。有了此类规划，购买了自动化测试

工具，并创建了自动化测试流程，但由于实施过程中遇到困难，没有继续推广解

决问题，而是过一段时间后，回到了原来的测试模式。

 实施了自动化测试，但却是失败的。公司实施了自动化测试，但是开发与

测试之间矛盾重重，对测试自动化的管理失控。这类组织虽然表面上还在勉强维持自动化测试，但实施的成本比手工测试增加了，工作量比从前更大了。这显然应该归于失败的范畴。

自动化测试实施的相对比较成功，但总是存在些问题。比如工具选择的不准

确，培训不到位，文档不完备，人员分配不合理，脚本可维护度不高等，从而造

成表面上的自动化测试流程，没有给测试工作带来实用的价值。

测试自动化实施的相对比较成功。对被测试产品功能进行了很好的前期评

估，使得相对比较稳定的功能实施了自动化测试，比较易于变化的功能采取手工

测试完成，这样自动化测试和手工测试试的结合给回归测试工作带来一定的方便并增加了测试覆盖度，同时便于测试管理。

因而从以上描述可以看出目前的自动化测试实施的还不够健全，因而自动化

 测试还不能完全代替手工测试，比较好的解决方案是自动化测试和手工测试相

 结合的方式。

**3.7几种典型的软件自动化测试框架**

在自动化的软件测试系统实现过程中使用框架设计可以使得测试脚本的维

护量减至最少。然而，大量的自动化测试工具均采用传统的“录制一回放”模

型，导致了较高的脚本维护量，因为测试数据在测试脚本程序中是以硬编码方

式实现的。此外，工具内建的测试用例除了测试应用程序的图形用户界面，实际

上并没有其它用处。因此，如何选择一个合适的测试自动化框架，是一个自动化

测试小组开始启动前需要最优先考虑的一个问题。

一个自动化测试框架就是一个由假设、概念以及为自动化测试提供支持的实

践的集合。以下描述五种基本的自动测试框架：模块化测试脚本框架，测试库构

架框架，关键字驱动/表驱动测试框架，数据驱动测试框架，以及混合测试框架。

可以根据实际需要去考虑采用其中的一种测试框架而不是仅仅依赖于一个简单

的捕获工具。同时，这些框架是了解自动测试框架以及根据自己的需要和经验来

设计自动测试框架的基础。

1．模块化测试框架

模块化测试脚本框架（TEST MODulARITY FRAMEWORK）需要创建小而独立的可以描述的模块、片断以及待测应用程序的脚本。这些树状结构的小脚本组合起来，就能组成能用于特定的测试用例的脚本。

在五种框架中，模块化框架是最容易掌握和使用的。在一个组件上方建立一

个抽象层使其在余下的应用中隐藏起来，这是众所周知的编程技巧。这样应用同

组件中的修改隔离开来，提供了程序设计的模块化特性。模块化测试脚本框架使

用这一抽象或者封装的原理来提高自动测试组合的可维护性和可升级性。

2．测试库框架

 测试库框架（Test Library Architecture）与模块化测试脚本框架很类似，并且具有同样的优点。不同的是测试库框架把待测应用程序分解为过程和函数而不是脚本。这个框架需要创建描述模块、片断以及待测应用程序的功能库文件。

3．关键字驱动或表驱动的测试框架

对于一个独立于应用的自动化框架，关键字驱动（KEYWORD DRIVEN）I9LJJ试和表驱动（TABLE DRIVEN）测试是可以互换的术语。这个框架需要开发数据表和关键字。这些数据表和关键字独立于执行它们的测试自动化工具，并可以用来“驱动＂待测应用程序和数据的测试脚本代码，关键宇驱动测试看上去与手工测试用例很类似。在一个关键字驱动测试中，把待测应用程序的功能和每个测试的执行步骤一起写到一个表中。

这个测试框架可以通过很少的代码来产生大量的测试用例。同样的代码在用

数据表来产生各个测试用例的同时被复用。

4．数据驱动测试框架

数据驱动（DATA DRIVEN），LJ试是一个框架。在这里测试的输入和输出数据是从数据文件中读取（数据池，ODBC源，CSV文件，EXCEL文件，ADO对象等）并且通过捕获工具生成或者手工生成的代码脚本被载入到变量中。在这个框架中，变量不仅被用来存放输入值还被用来存放输出的验证值。整个程序中，测试脚本来读取数值文件，记载测试状态和信息。这类似于表驱动测试，在表驱动测 试中，它的测试用例是包含在数据文件而不是在脚本中，对于数据而言，脚本仅仅是一个“驱动器”，或者是一个传送机构。然而，数据驱动测试不同于表驱动测试，尽管导航数据并不包含在表结构中。在数据驱动测试中，数据文件中只包含测试数据。

这个框架意图减少需要执行所有测试用例所需要的总的测试脚本数。数据驱

动需要很少的代码来产生大量的测试用例，这与表驱动极其类似。

 5．混合测试自动化（Hybrid Test Automation）框架

 最普遍的执行框架是上面介绍的所有技术的一个结合，取其长处，弥补其不

足。这个混合测试框架是由大部分框架随着时间并经过若干项目演化而来的。

**3.8自动化测试工具的比较**

目前国内比较流行的自动化测试工具多种多样，例如Ratianal公司的自动化测试工具Functional Tester，HP Mercury的功能测试工具WinRunner，QTP，ThoughtWorks公司开发的开源工具Selenium，性能测试工具LoadRunner，Compuware公司的QA Run，SegueE公司的Silk Test，Silk Perfoemance等。

对以上一些自动化测试工具进行研究我们就会发现每个工具都有其支持的

侧重点，尤其对被测试软件中各种控件的录制和识别，需要工具进行严格的支持。

所以在进行～项自动化测试之前，对测试工具的选型显得尤为重要。一般地，对

于自动化测试工具具有如下要求：

支持脚本化语言：包括支持数组、列表、结构等多种常用的变量和数据类型、

各种条件逻辑和循环、函数的创建和调用。如果此工具使用如C#、Java等通用语言，测试就更方便。脚本语言的功能越强大，就能为测试者提供更灵活的使用空间，或者用它写出比被测软件还要复杂得多的测试系统。

对程序界面中对象的识别能力：工具必须能够将程序界面中的所有对象都区

分井标识出来，录制的测试脚本才具有更好的可读性、灵活性和更大的修改空间。

支持函数的可重用：用它可以建立一套比较通用的函数库，一旦程序做了修

改，只需把原脚本中的相应函数进行更改，而不用把所有可能的脚本都改动，可

以大大节省工作量。

支持外部函数库：一些外部函数同样能够为测试提供更强大功能，如

Windows程序中对DLL文件的访问，Client/Server程序中对数据库编程接口的调用等。

抽象层：可以将程序界面中存在的所有对象实体一一映射成逻辑对象，测试

就针对逻辑对象进行，这样当程序界面改变时，就可大大减少测试维护工作量。

分布式测试支持：一个测试项目，通常需要多名测试人员协同工作，因此如

果测试工具不支持分布式测试，将很难保障这种协同测试工作的开展。分布式测

试最大的好处是我们可以事先定制任务执行的时间表，如在指定时间、指定设备

上执行指定测试任务。

支持数据驱动测试：在数据驱动测试中，只需编制少量的脚本，而准备大量

的测试数据，测试工具可以自动重复完成大量的测试工作。比如在Windows系统中，如果测试工具支持对INI文件的处理，被测软件通过读INI文件知道要在哪一台服务器上运行，那么我们只要把需要运行程序的服务器名写进INI文件

中，而不需要修改测试脚本，就可以让程序在指定设备上运行。

错误处理：利用它可以避免测试程序因一些异常错误而异常终止。这样就可

将测试任务提前定制好，在不需要很多人为关注的情况下完成测试脚本执行和结

果记录工作。

软件配置管理：可以帮助我们进行测试脚本库的倒入、倒出，回退到以前版

本、比较不同版本间差别，以及同时对几个项目进行跟踪等，尤其在团队开发中

很有必要，可以对测试数据文件、测试脚本、对象抽象层进行统一管理。如果测

试环境与开发环境使用同一套版本管理工具，能够给项目管理带来更大的便利。

支持脚本的命令行方式执行：如果能够通过命令行方式运行测试脚本，可以

为测试的执行带来更大的灵活性。如机器启动时、程序BUILD后都可以自动启动测试脚本的运行。

ThoughtWorks开发的开源功能自动化测试工具Selenium，不仅满足以上自动化测试工具的所有要求，而且支持多种操作系统，如Windoes，Mac，Linux，也支持多种浏览器，如IE，FireFox，Chrome等。而且它可以使用常用的面向对象语言如C#，Java语言作为其自动化测试脚本，使其具有更大的灵活力及表达力，而且Selenium支持多个测试用例并行执行，大大提高了自动化测试的效率。所以我们针对Selenium自动化测试中出现的问题进行重点的研究。

**4 构建及使用自动化测试框架**

**4.1 Selenium简介**

Selenium是一个主要由ThoughtWorks公司开发的开源Web应用系统功能测试工具。Selenium的测试脚本直接运行在浏览器中，支持Windows平台上的IE，FireFox，Chrome等浏览器以及Windows和Linux平台。

**4.1.1Selenium WebDriver**

Selenium WebDriver是WebDriver和Selenium Rc的合并成果。目前，WebDriver支持的语言绑定包括Java、C#、Python和Ruby。它支持Chrome、Firefox、Opera和Android、iPhone浏览器。​此外，还有其他关联项目，不在同一源代码库中维护，但是和主项目（Selenium WebDriver）紧密合作，例如提供Perl绑定支持、BlackBerry浏览器支持，以及“无头”WebKit——用于持续集成的测试其无法正常显示的情况。最初的Selenium RC机制仍然维持，帮助WebDriver在浏览器不受支持的情况下提供支持。

**4.2 Selenium WebDriver框架的建立**

Selenium WebDriver的框架也就是基于selenium+junit的框架。配置

JDK1.6环境，安装MyEclipse,在FireFox浏览器中安装selenium IDE,接下来配置MyEclipse与Junit。

1. 在主菜单栏上选择【窗口->首选项】，出现首选项对话框

2. 展开java节点，选择【构建路径->类路径变量】：

3. 点击新建按钮，在对话框中输入新的变量名JUNIT，设置路径为junit.jar，可以在安装目录/eclipse/plugins/org.junit\_3.8.1/junit.jar下找到junit压缩包。

4. 为了debug的需要，还要添加JUint包的源代码，可以在MyEclipse安装目录/eclipse/p;ugins/org.eclipse.jdt.source\_3.0.2/下搜索到junitsrc.zip。为JUnit源代码创建一个新的变量JUNIT\_SRC，按照上面的步骤将其连接到junitsrc.zip所在的路径。

 框架搭建好了，下面看看怎样使用。

**4.2.1 录制和回放**

打开selenium，出现的页面如下图。

**图4.2.1-1**  **selenium界面**

执行功能测试用例并录制脚本。我选择了一个简单的例子，登录新浪微博。

1、打开新浪首页；

2、进入新浪微博；

3、出现登录页面输入用户名和密码；

4、输入验证码，进入微博。

上面的四步操作都用selenium进行录制，录制完毕后的脚步如下图示，如果需要的话可以在这里进行一些编辑，一个非常有用的功能是在Target里输入页面元素的xpath，点击Find按钮。如下图所示：

**图4.2.1-2** **selenium录制脚本**

录制完之后可以点击Table上面的按钮进行回放、回放可以通过调速来观看每一步的操作。

测试用例转换为自动化脚本 只在Selenium IDE里运行是远远不能满足[需求](http://www.ltesting.net/ceshi/ruanjianzhiliangbaozheng/xqgl/)的，我们需要为测试用例增加一些特性以使其能自动运行持续[回归](http://www.ltesting.net/ceshi/ceshijishu/csgl/hgcs/)运行于不同的浏览器上，而Selenium IDE可以方便实现用例转换为自动化脚本，而且支持多种语言，如下面两张图片所示：

**图4.2.1-3** **代码转换**

**4.2.2 录制、编辑和回放**

<html >

<head >

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link rel="selenium.base" href="http://www.baidu.com/" />

<title>New Test</title>

</head>

<body>

<table cellpadding="1" cellspacing="1" border="1">

<thead>

<tr><td rowspan="1" colspan="3">New Test</td></tr>

</thead><tbody>

<tr>

 <td>open</td>

 <td>/s?wd=selenium&amp;rsv\_bp=0&amp;ch=&amp;tn=06008006\_2\_pg&amp;bar=&amp;rsv\_spt=3&amp;inputT=3040</td>

 <td></td>

</tr>

<tr>

 <td>click</td>

 <td>link=Selenium教程1-IDE的基本使用\_百度文库</td>

 <td></td>

</tr>

<tr>

 <td>click</td>

 <td>link=Selenium - 集成测试工具 - 开源中国</td>

 <td></td>

</tr>

<tr>

 <td>click</td>

 <td>link=selenium\_百度百科</td>

 <td></td>

</tr>

<tr>

 <td>clickAndWait</td>

 <td>link=下一页&gt;</td>

 <td></td>

</tr>

<tr>

 <td>click</td>

 <td>id=container</td>

 <td></td>

</tr>

</tbody></table>

</body>

</html>

上面是录制的一段代码将代码导入到MyEclipse中可以进行编辑和回放的效果。

**4.2.3编程和回放**

 下面编写一段简单的代码进行回放

@Test

 public void testSearchSuccess(){

 driver.get( "http://www.google.com" );

 driver.get("http://www.51testing.com/#hl=zh-CN&newwindow=1&safe=strict&site=&source=hp&q=selenium+driver%E6%8E%A5%E5%8F%A3&oq=selenium+driver%E6%8E%A5%E5%8F%A3&aq=f&aqi=&aql=&gs\_l=hp.3...832.25011.0.25453.57.38.0.0.0.0.0.0..0.0...0.0.wyE18SyGKJY&bav=on.2,or.r\_gc.r\_pw.,cf.osb&fp=3ad321088b6de2e5&biw=1366&bih=618");

 driver.findElement(By.*linkText*("Selenium - 51Testing软件测试网")).click();

 driver.findElement(By.*linkText*("我的足迹"));

上面的代码就是运行时打开Google，搜索selenium，得到的搜索页面，点击进入[www.51testing.com-51test](http://www.51testing.com-51test)软件测试网，在点击我的足迹。

**4.2.4 数据驱动测试**

我们从一个最简单的登录例子开始。

1.最开始我们需要验证在用户名和密码都正确的情况下，能够正常登录系统，我们这样编写测试代码：

@Test

def should\_login\_success\_with\_exist\_username\_and\_correct\_password(){

 LoginPage page = user.open(LoginPage,"/login.html")

 user.perform("login",['user1','1234'],on(page))

 assert page.successLogin

}

运行一下，出现红条。为什么呢？原来测试数据库里没有用户名为user1的用户，写个数据库数据初始化脚本。再运行，好了，绿条。

接下来我们再增加一个测试，需要覆盖密码错误时不能登录系统的情况，很快测试就完成了：

@Test

defshould\_login\_success\_with\_exist\_username\_and\_incorrect\_password(){

 LoginPage page = user.open(LoginPage,"/login.html")

 user.perform("login",['user1','4321'],on(page))

 assert page.successLogin,false

}

再运行一下测试，绿条。现在可以看下这段代码，有些重复，重构一下：

@Test

def should\_login\_success\_with\_exist\_username\_and\_correct\_password(){

 assert login('user1','1234')

}

@Test

defshould\_login\_success\_with\_exist\_username\_and\_incorrect\_password(){

 assert login('user1','4321'),false

}

def login(username,password){

 LoginPage page = user.open(LoginPage,"/login.html")

 user.perform("login",[username,password],on(page))

 return page.successLogin

}

重构完成，可以看到，我们的测试方法里现在没有了任何行为，仅仅是数据！感觉有点怪，先用TestNG提供的@dataProvider整理一下：

@Test(dataProvider="testdata")

def testLogin(username,password,expected){

 LoginPage page = user.open(LoginPage,"/login.html")

 user.perform("login",[username,password],on(page))

 assert page.successLogin,expected

}

@DataProvider(name="testdata")

def Object[][] dataForLogin(){

 def data=new Object[2][]

 data[0]=['user1','1234',true] as Object[]

 data[1]=['user1','4321',false] as Object[]

}

测试方法只剩下了一个！如果要测试不存在的用户不能登录系统呢？很简单，增加数据即可！

@DataProvider(name="testdata")

def Object[][] dataForLogin(){

 def data=new Object[2][]

 data[0]=['user1','1234',true] as Object[]

 data[1]=['user1','4321',false] as Object[]

 data[1]=['inexistuser','1234',false] as Object[]

}

在我们的测试方法里，测试数据和测试的行为进行了完全的分离。从系统的功能来说，功能一旦实现，那么就是一个黑盒，我们只要提供数据即可进行测试，这个数据包括两部分：输入和期待的输出。这些方式为什么会存在呢？因为这些方法是为了覆盖功能的各个路径的，是提高测试覆盖率的。那么为什么会产生这么多的测试方法呢？那是在这些测试方法里，测试数据和测试行为是耦合在一起的。

突然想到 QA想增加测试用例只需要自己增加数据就可以了。看来看去发觉某个功能实现有问题。差了相关资料发现因为数据库变了，数据变了，应用服务器变了，所以会有些问题。登录到QA环境，一测试，还真是。

怎么办？修复完BUG，第一反应就是自动化测试能不能跑在QA环境呢？一般情况下，这些测试需要干净的测试环境，我们会制造很多的测试数据，可是在QA环境下，QA有它自己的测试数据，这些数据都不存在了。恩，看看刚才的测试代码，用QA的数据也可以，于是把代码改了：

@DataProvider(name="testdata")

def Object[][] dataForLogin(){

def data=new Object[2][]

 data[0]=['hrong','1234',true] as Object[]

 data[1]=['hrong','4321',false] as Object[]

 data[1]=['rhao','1234',false] as Object[]

}

可以了！为了该测试既能在开发测试环境运行又能在QA环境下运行，我们可以引入一个环境变量，将测试数据放到文件里，通过环境变量来加载不同的测试数据。

**4.2.5 测试用例的编写**

测试用例的编写主要有三种情况：

1、happy-path主干路径测试即输入正确的数据；

2、异常情况即输入空格或特殊字符；

3、than边界情况如测试字符过长。

**5 总结与展望**

**5.1总结**

软件测试一直是软件工程不断研究发展的领域，特别是随着Web应用系统

的发展及Web2.0的到来，用户对Web应用系统的体验越来越重要，所以基于Web应用系统的功能测试特别重要。随着系统的复杂程度越来越大，越来越需要自动化的测试去完成那些重复的，机械的测试工作，让测试人员完成更多需要智力活动的测试任务，以提高测试的效率。

本文通过对当前基于Web应用系统的功能测试进行了讨论，根据实际测试

中出现的测试脚本维护量大的问题，提出了一个将元素的定位信息抽取，并通过

元素的定位逻辑来动态定位被测页面元素的解决此方案。SeleniumWebDriver运行模式，采用Java语言来实现测试脚本的编写，并实现了根据元素请求参数能过元素描述配置文件得到元素定位信息的过程。

**5.2展望**

目前此解决方案还需要部分时间去维护一个页面所有元素的描述信息的置文件，如果一个网站的被测的页面没有整体的统一风格或者一个项目中需要被

测试的页面数量很多的时候，需要维护的元素描述信息就会非常庞大，不论是新

建，查找相关元素，还是维护这个配置文件都需要很多的时间，这样整体测试用

例的成本就会变大。如果可以将这些庞大的元素描述信息存入数据库，如果数据

库的结构设计良好，再设计一个Gui程序以树状或者其它直观的方式显示元素

描述信息的结构，方便测试人员去添加，修改这些元素描述信息，这样就可以减

少维护元素描述配置文件信息的工作量，提高测试的效率。

另一个问题是，本方案只是基于Selenium开源功能测试工具，并用Java语言来实现的，其元素描述中的定位逻辑采用JavaScript编写，对Selenium的依赖性很强，如何改进此方案，使其它功能测试工也可以使用该描述配置文件，也是一个需要继续研究的问题。

**致谢**

历时将近两个月的时间终于将这篇论文写完，在论文的写作过程中遇到了无数的困难和障碍，都在同学和老师的帮助下度过了。尤其要强烈感谢我的论文指导老师，她对我进行了无私的指导和帮助，不厌其烦的帮助进行论文的修改和改进。另外，在校图书馆查找资料的时候，图书馆的老师也给我提供了很多方面的支持与帮助。在此向帮助和指导过我的各位老师表示最中心的感谢！

感谢这篇论文所涉及到的各位学者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。

感谢我的同学和朋友，在我写论文的过程中给予我了很多你问素材，还在论文的撰写和排版灯过程中提供热情的帮助。

由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正！

参考文献

[1]包晓露 于秀山 软件测试自动化技术的研究与实践[M]．军队指挥自动化 2002年4月

[2]姚砺，束永安．软件测试自动化关键技术的研究[M]．安徽大学学报（自然科学版）

2003年3月

[3]张丽波．软件自动化测试的设计与实施[M]．佳木斯大学学报（自然科学版）2004年4月

[4]张振兴．浅谈软件测试自动化解决方案[M]．<http://www.soft6.com/tech/4/46079.html>

2005年9月

[5]宋波，张忠能．基于系统功能测试的软件自动化测试可行性分析[M]．计算机应用与软件 2005年12月

[6]朱菊，王志坚，杨雪．基于数据驱动的软件自动化测试框架[M]．计算机技术与发展

2006年5月