Software Testing

软件测试丛书

软件自动化测试框架设计与实践

Software Automation Testing Framework Design

柳胜 编著

- 集人员组织、流程管理、测试技术于一体的自动化测试框架实战知识，引导读者成功实施自动化测试
- 基于QTP的自动化测试框架搭建，制定脚本规范和框架接口标准
- 自动化测试框架实例—Automation Center（AC），涵盖测试框架结构设计与应用开发技术
- 网站www.cesoo.com/bbs提供本书源代码及相关视频文件下载
- 资深测试专家提供在线答疑

人民邮电出版社
POSTS & TELECOM PRESS

软件测试丛书

软件自动化测试框架设计与实践

Software Automation Testing Framework Design

柳胜 编著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

软件自动化测试框架设计与实践 / 柳胜编著. -- 北京 : 人民邮电出版社, 2009.11
ISBN 978-7-115-21513-0

Ⅰ. ①软… Ⅱ. ①柳… Ⅲ. ①软件－测试－自动化 Ⅳ. ①TP311.5

中国版本图书馆CIP数据核字(2009)第186148号

内 容 提 要

本书从自动化测试思想、技术和实施操作等层面进行深入分析，全面讲解了如何针对企业或项目需求，并以量体裁衣的方式来设计完成自动化测试框架，从而为自动化测试实施的企业和个人提供实战指南。

本书分 3 篇，第一篇初级篇，包括第 1～4 章，主要介绍自动化测试的基础知识和经验，以及自动化测试团队的建设等；第二篇中级篇，包括第 5～7 章，主要介绍自动化测试框架的构建思想，以及在 UI 测试自动化和单元测试自动化等领域内的技术实现；第三篇高级篇，包括第 8～9 章，主要讲解自动化测试框架的实例开发过程，开发高质量的实例代码等内容。

本书旨在帮助读者学习和理解测试框架的设计原则和实施技巧，以便根据测试的项目特点贯彻实施。读者阅读本书之后，将会理解并把握如何根据项目和产品特点构建一个高效、高质量的自动化测试解决方案。

软件自动化测试框架设计与实践

◆ 编　　著　柳　胜
责任编辑　魏雪萍
执行编辑　张　涛

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：15.5
字数：337 千字　　2009 年 11 月第 1 版
印数：1－3 500 册　　2009 年 11 月北京第 1 次印刷

ISBN 978-7-115-21513-0

定价：45.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前　言

自动化测试是当今软件测试行业一个很受关注的方向。目前，业界实施比较成熟的是性能测试自动化。在系统测试自动化方面，包括 UI 测试自动化、单元测试自动化等领域，虽然可以用强大的第三方测试工具作为解决方案，如 QTP、WinRunner、Selenium 等，但在具体实践中，企业依靠某个工具很难成功实施测试自动化。究其原因，一是，理论上利用测试工具进行自动化测试可以替代部分工作，但在实践过程中并不能完整地解决测试质量与效率、投入与产出等一系列现实因素之间存在的矛盾。从这个角度来说，测试工具提供的测试技术只是自动化测试实施的一部分。二是，测试人员开发了一堆自动化测试脚本若没有具体的框架来规定实施和执行的原则，也会无用武之地，这就像造车一样，只具备了零件等元素但缺少整体的造车图纸来指导使用它们也难以造出整车。因此，现实的测试实践中，我们需要一个完整而务实的自动化测试解决方案——测试框架，以便真正地实现高效、高质量的软件自动化测试。

测试框架（Test Framework）作为实现高效率、高质量自动化测试的完整解决方案，从诞生之日开始，越来越多的软件组织和个人用自己的逻辑去诠释测试框架，所以，我们听到了种种说法，一套测试管理系统被称之为测试框架，一个测试工具被冠以关键字驱动框架之名，甚至，一段程序也被声称其实现了数据驱动的框架。测试框架犹如盲人摸象中的那头大象一样，有人说它是一个软件，只不过它的功能是测试另外一个软件，有人认为它是一套流程和规范，否则怎称框架。

本书的作者看来，所谓“测试框架”这个概念只是一个封装了很多东西的盒子，这个盒子的外观和形状对我们来说无关紧要，我们最关心的是这个盒子里面到底存放了什么东西，否则就成了“买椟还珠”的现代版。因此，本书旨在帮助读者打开测试框架这个盒子，把里面的“宝贝”呈现给读者。相信读者读了本书之后，不会再追问测试框架到底是什么，而会更关心怎样构建一个高效务实的自动化测试解决方案。

本书从自动化测试思想、技术和实施操作等层面进行了深入分析，全面讲解了如何针对企业或项目需求，并以量体裁衣的方式来设计完成自动化测试框架，从而为自动化测试实施的企业和个人提供实战指南。具体内容包括自动化测试实施风险因素分析、PEARL 过程模型建立、自动化测试团队的培养和规划、自动化框架设计思想、自动化测试技术储备、自动化测试工具的有效评估、高质量程序/脚本编程技巧与原则等。另外，本书并不停留在框架理论层面，在第 8 章介绍了框架实例分析——Automation Center（AC），这些都能对从事自动化测试的读者有很强的现实借鉴意义。

本书特色

本书全面阐述了集测试人员组织、流程管理、开发技术于一体的自动化测试框架的主要知识，并通过大量实例贯穿每个知识点，对于开发框架的核心知识反复强调和应用，真正做到了学用结合地引导读者掌握自动化测试框架开发的技术，是软件测试业界第一本有关自动化测试框架开发的实战技术书籍。

谁适合阅读本书

本书面向有一定自动化开发经验的读者，通过本书的学习，可以迅速掌握如何根据实际测试需求，设计和实现自动化测试框架。对于刚入门自动化测试的读者也可通过本书的学习，树立正确的自动化测试思想，找到测试实战的捷径，尽快融入实战角色。

本书特别适合以下类型的读者：

- 希望学习软件测试框架知识的初级、中级、高级测试人员；
- 希望解决应用测试框架过程中遇到问题的设计、执行、分析等相关人员；
- 测试组长、测试经理、质量保证工程师、软件过程改进人员。

阅读建议

本书按照循序渐进的思路安排内容，无论读者是有经验的软件测试人员、开发人员、系统管理人员还是刚参加工作的测试人员，建议最好按照目录顺序进行阅读。因为全书是按照概念解析、实践应用、问题解答的顺序来编写的，不仅仅在实践和问题解答部分提供了很多真实、详细的案例，在进行概念解析的同时也提供了很多非常重要的经验，这对于读者深入理解相关概念和少走弯路都是不无裨益的。当然，对于自动化测试经验相对丰富的读者也可以依据自己的需要，选择关心的内容进行针对性的阅读。

本书约定

本书遵循如下约定。

符号和术语	含　义	示　例
【案例】	每章节中对应的实战案例	例如，案例 6，内存的利用和回收
【名词解释】	对关键术语的解释	例如，名词解释：HTML DOM
【解析】	表示对文中的概念和术语进一步给出拓展说明	例如，一个软件产品提供多国语言界面。我们可以构想这样的自动化测试：录制一份脚本，然后在多个语言界面下运行，只要我们能够得到每个页面上的 text 的翻译
【思考】	是对已讲解知识的延伸，以便让读者在思考中学习	例如，如果是基于白盒的单元自动化测试，即代码是可见的，应该如何设计测试案例？
【注意】	关键知识点总结	在上面的示例里，并没有桩代码，想想这是为什么？URLEncoder 的桩是什么？

本书作者

本书编著柳胜，计算机应用硕士，拥有多年的软件开发和测试实践经验。尤其擅长在自动化测试工具应用、性能测试和单元测试等方面的工作。曾在摩托罗拉等大型外企担任高级开发工程师、高级自动化测试工程师等职务。

本书编委

张海波、魏岩岩、王伟庆、赵子如、马志强、王智群、郑金鑫、李东方、王冰、王颖、于迪、厉敏、刘桂梅、厉德仁、田月琴等。

网上答疑

在本书写作过程中，本人已尽力而为，但由于时间仓促，水平有限，书中难免有错误之处，如读者在阅读过程中，发现本书存在错误或不妥之处，欢迎与作者联系，以便作者及时更正。本书的源代码、开发教学视频、勘误、更新信息、答疑信息都可以从作者的博客（http://www.cesoo.com）上直接获得。读者如有疑问，可以访问作者的博客直接留言，也可以和本书责任编辑联系，联系邮箱为 zhangtao@ptress.com.cn。

致谢

在本书写作过程中，很多测试同行提供了宝贵建议，在这里向他们表示衷心的感谢！

编者

目　录

第一篇　初级篇——认识自动化测试框架

第 1 章　将降大任——自动化测试 ……1
1.1　软件测试面临的困境与迷局 ……2
1.1.1　软件质量困境 ……2
1.1.2　软件成本困境 ……6
1.2　“时势造英雄”——软件测试自动化的异军突起 ……8
1.2.1　软件自动化测试如何代替手工测试 ……8
1.2.2　推动软件自动化测试的源动力 ……12
第 2 章　“神话”破灭——自动化测试能否担当大任 ……16
2.1　企业自动化测试实施的情景 ……17
2.1.1　自动化测试实施背景介绍 ……17
2.1.2　自动化测试实施场景回放 ……17
2.2　自动化测试的“神话”破灭 ……20
2.2.1　昂贵的自动化测试实施成本 ……20
2.2.2　实际上并不强大的自动化测试脚本 ……21
2.2.3　自动化测试实施的命门：维护成本 ……22
第 3 章　成功之道——如何构建高质量的自动化测试 ……24
3.1　PERAL 模型的实施背景 ……25
3.2　Purpose：如何建立一个务实明确的自动化测试目标 ……25
3.2.1　问题 1：自动化还是手工测试 ……26
3.2.2　问题 2：如何估算分析自动化测试效益 ……26
3.2.3　问题 3：如何构建高收益成本比自动化测试目标 ……29
3.3　Evaluation：评估和估算的量化决策指南 ……37
3.3.1　工具选择定律一：测试界面决定工具类族 ……38
3.3.2　工具选择定律二：测试项目综合特征确定工具应用方案 ……41
3.3.3　工具选择实际案例分析 ……44
3.3.4　总结 ……47
3.4　Architecture：构建和设计自动化测试 ……48
3.4.1　自动化测试的最终用户是测试工程师 ……48
3.4.2　自动化测试的实质是开发一个测试软件 ……49
3.5　Run and Debug：开发调试 ……49
3.5.1　高内聚和低耦合的模块实现原则 ……49
3.5.2　数据驱动原则 ……53

3.5.3 自动化脚本开发质量优先级……56
3.6 Link with Manual test：自动化测试与手工测试的有效整合……57
3.6.1 自动化测试和手工测试的关系……58
3.6.2 自动化测试与手工测试流程整合……59
3.7 PEARL 模型实施成功经验……60
3.7.1 重置目标，长远规划……60
3.7.2 强大的自动化测试框架…61
3.8 总结……62
第 4 章 组织实施——怎样建立与培养自动化测试团队……63
4.1 测试团队简介……64
4.2 明确自动化测试目标……64
4.2.1 好的目标是自动化测试实施的发动机……64
4.2.2 建立一个高收益并可行的自动化测试实施目标…65
4.3 积极有效的沟通技巧……70
4.3.1 勤汇报，多交流……72
4.3.2 实用为先……74
4.4 培养和建立自动化测试团队……75
4.4.1 手工测试团队的规划……76
4.4.2 自动化测试实施中的团队……78
4.4.3 自动化测试实施后的团队……80
4.5 自动化测试技术储备……81

第二篇 中级篇——自动化测试框架基本原理及实现

第 5 章 庐山真面目——自动化测试框架……82
5.1 自动化测试框架简介……83
5.2 测试的自动化——以工具为中心……83
5.3 百家争鸣——形形色色的自动化测试框架……85
5.3.1 数据驱动测试框架（The Data-Driven Testing Framework）……85
5.3.2 关键字驱动或表驱动测试框架（The Keyword-Driven or Table-Driven Testing Framework）……89
5.3.3 总结……90
5.4 自动化的测试——测试框架原型……91
5.4.1 框架包括的具体技术……91
5.4.2 框架的用户……92
5.4.3 制定和开发框架……93
5.5 测试框架集大成者——无需人工干预的自动化回归测试……93
第 6 章 实例研究——单元自动化测试框架解决方案……95
6.1 被测对象介绍……96
6.1.1 背景简介……96
6.1.2 单元测试对象 URLEncoder.encode 函数介绍……97
6.1.3 对 URLEncoder.encode 的单元测试案例设计……98
6.2 自动化测试框架逐步实施……100
6.2.1 第一步：单元测试自动化……100
6.2.2 第二步：框架——数据

驱动……101
6.2.3 第三步：框架——整合开发测试流程……104
6.2.4 第四步：框架（高级）——定义自动化测试管理策略和规范……107
6.3 单元自动化测试框架实施总结……108
第 7 章 实例研究——基于 UI 功能的自动化测试框架解决方案……109
7.1 GUI 的软件自动化测试原理与技术基础……110
7.1.1 GUI 软件测试简介……110
7.1.2 GUI 自动化测试原理与实例演示……114
7.1.3 在构建自动化测试框架时的工具因素……124
7.2 基于 QTP 的功能自动化测试框架原型的搭建……125
7.2.1 QTP 工具简介……125
7.2.2 自动化测试框架预期功能目标……126
7.2.3 自动化测试框架的实现……127

第三篇 高级篇——自动化测试框架案例实战

第 8 章 自动化测试框架实例——Automation Center（AC）……142
8.1 产品测试案例分析……143
8.1.1 被测软件产品介绍——某大型分布式企业协同组件介绍……143
8.1.2 软件产品测试需求分析……144
8.1.3 测试需求矩阵分析……146
8.1.4 全球化测试需求分析……149
8.2 测试中应用 AC 的效益分析……153
8.2.1 预期成本计算……154
8.2.2 预期收益计算……154
8.2.3 预期收益比……155
8.2.4 AC 实施过程规划……155
8.3 AC 框架中脚本开发规范……156
8.3.1 自动化测试中的规范……156
8.3.2 规范应该考虑的因素……157
8.3.3 有效地推行自动化测试规范……159
8.3.4 AC 中 QTP 脚本规范实例……159
8.4 AC 框架测试报告格式及规范……163
8.4.1 总览报告……163
8.4.2 细分报告……164
8.4.3 AC 测试报告总结……169
8.5 AC 框架中客户端自动化测试解决方案……171
8.5.1 AC 中 QTP Agent 介绍……171
8.5.2 “一次编码，多语言运行”的 QTP 脚本开发思想及实现……172
8.5.3 AC 中 QTP Agent 在企业内部环境的部署……178
8.5.4 AC 中 QTP Agent 收益……183
8.6 AC 中安装自动化解决方案……185
8.6.1 Windows 下软件安装：默认式安装与交互式安装……185

8.6.2 Linux 下软件安装：文本模式与图形化模式……192
8.7 AC 中自动化管理与控制平台……199
8.7.1 AC 拓扑结构设计……199
8.7.2 AC 的工作协作图……202
8.7.3 AC 工作流程图……203
8.8 AC 应用经验……205
8.8.1 明确而务实的需求……205
8.8.2 有效的人员培训……205
8.8.3 敏捷高效的反馈机制……206
8.9 AC 开发技术实例……206
8.9.1 基于事务（transaction）机制的测试案例状态栈的实现……206
8.9.2 测试案例状态轮询 Java 实现……207
8.9.3 AC 与 Agent 协议原语……208
第 9 章 开发高质量测试脚本……210
9.1 案例 1：脚本开始处首先进行环境检查……211
9.2 案例 2：函数入口检查……213
9.3 案例 3：使用正则表达式处理字符串……214
9.4 案例 4：脚本健壮性出错处理……219
9.5 案例 5：测试脚本中参数变量的规范命名……222
9.6 案例 6：内存的利用和回收……225
9.7 案例 7：数据驱动……228
附录 与自动化测试实施人员的对话实录……232

第一篇 初级篇
——认识自动化测试框架

Chapter 1

第1章 将降大任——自动化测试

在软件测试日新月异发展的今天，测试自动化正在成为软件测试中的一颗非常受瞩目的新星，短短的几年内，各种自动化测试的工具和技术层出不穷，软件组织和个人也对自动化测试推崇备至，乐此不疲地实践应用和学习。但让测试从业者尴尬的是，自动化测试在实用之后，回报给我们的是巨大的先期开发投入和繁琐的后期维护陷阱，更像是一个吃得多吐得少的黑洞，不由得让人对此心生疑虑，自动化测试到底是在软件测试中昙花一现？还是一颗稳定闪耀的恒星？如果是恒星，它的动力来自哪里？自动化测试成功之路又从何处开始呢？

1.1 软件测试面临的困境与迷局

如果想知道自动化测试这颗种子是否能长成参天大树，那么我们就必须把它放到其生长环境里去观察。这就是我们所说的“时势”。

每种物质运动都有其自己的“时势”，它是如此的重要和强大，但却偏偏隐居幕后，像一只看不见的强有力的大手，掌控着万物的衰荣。在现实中，我们看到成功者之所以成功，往往不是因为付出了很多的努力（付出努力的人比比皆是），而是因为他在合适的时间和地点，坐在了一个合适的位置上，结果就像等公交车一样，“时势”路公交车把他带向了成功的终点站。一位伟人曾经说过“时来天地皆同力，运去英雄不自由”指的就是这个道理。

对软件自动化测试来说，我们要研究测试自动化的“时势”，就要“风物长宜放眼量”，把眼光放到软件测试整个行业发展趋势中去。从实践中去看自动化测试的位置和作用，然后推断出自动化测试的发展空间。

软件行业有着怎样的现状和趋势呢？下面我们将进行分析。

1.1.1 软件质量困境

随着软件对社会的影响越来越大，用户对软件质量的要求越来越高，软件质量困境表现得越来越突出。

软件产业虽然只有短短几十年的历程，但是其应用范围已经从最初的科研专用转变为渗透到我们生产生活的各个方面，起着非常重要的作用。比如，我们出门坐公交车，网上订餐、购物，物业信息化管理等，这些都是应用软件起作用的。人类社会对软件的依赖正在越来越强，反过来，软件质量问题对人类社会的影响也越来越大。这种例子在当今社会生活中比比皆是，例如，2008 年发生的大型订票网站在开通首日即因访问流量过大停工的问题，导致上百万人购票失败；某银行黄金交易系统出现漏洞，两名非法入侵者通过低买高卖获利 3000 多万元等。这对于智慧的人类自身来说，真是一个莫大的嘲讽。在以前，谁能想到因软件问

题会让人买不到票，乘不上车，吃不成饭呢？

因此，在软件影响人类社会各个方面（除了决策权）且已经成为趋势的前提下，我们不得不加倍小心防范软件问题带来的危害和损失，确保一些问题能够在软件上线投入使用之前被发现和解决，就需要加强软件质量管理的意识。

那么软件企业如何确保软件质量呢？通行的做法是软件测试。对于国内外大多数软件企业来说，软件测试仍是保证软件质量最重要和有效的手段。（软件质量问题解决办法有两种，一种是“未雨绸缪”的前期软件过程控制，一种是“亡羊补牢”的事后补救。软件测试属于后者，但由于“未雨绸缪”的软件过程控制在软件行业中还不成熟，相比来说，软件测试更为实用和有效）。

对于软件测试来说，想要达到提高软件质量的目的，就要找到尽可能多的软件 Bug。这做起来其实没有什么秘笈，只有两种办法，“多测”和“测多”。

“多测”就是测试周期变短，测试执行频率加快。以前一次产品发布（release）发行测一轮，现在一个版本（build）建立就要测一轮，这样做的目的是要在第一时间发现 Bug 并解决，从而减少修复 Bug 的成本。

“测多”就是每次软件测试执行要运行更多的测试案例，覆盖更多的功能模块，保证每个角落不会有遗漏，发现尽可能多的 Bug。

下面我们详细地分析一下“多测”和“测多”策略的根源所在。

1. Bug 衍生模型 —— “多测”

下面我们来看一下 Bug 是如何产生的。在瀑布模型下，我们通常把软件的开发阶段在时间上划分为需求分析阶段、设计阶段（又可细分为概要设计和详细设计）和编码阶段，如图 1-1 所示。

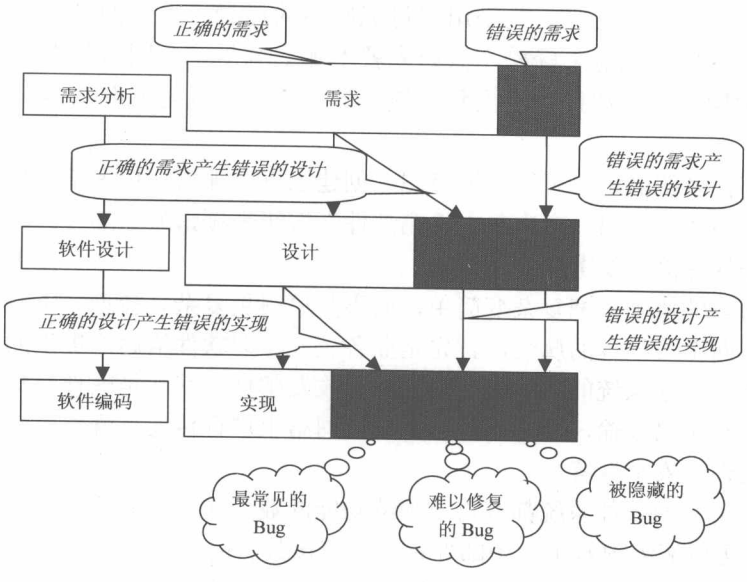

图 1-1　Bug 衍生模型

在图 1-1 中我们可以看到，究其根本，Bug 的来源有两种，一类是本阶段内产生的 Bug，比如在进行需求分析的时候，产生了一个错误的需求；另外一类是本阶段向下一阶段转化迁移过程中产生的 Bug，比如设计阶段向编码阶段转化的过程中，正确的设计产生了错误的编码实现。

从图 1-1 中，我们还可以发现一个有意思的事实，如果需求从一开始就错了，那么，这个因错误需求而导致错误实现的 Bug（如颜色最深的 Bug 部分）在软件测试阶段是不可能被发现的（被隐藏的 Bug），一直到软件 release 到客户那里，才可能被客户发现，这时修复 Bug 的代价非常惊人（除了客户的损失之外，至少要有 Bug 修复、全回归测试、补丁测试等）；而如果软件设计错了而导致错误的实现，这样的 Bug 称之为难以修复的 Bug（颜色较深的 Bug 部分）。比如，让众多开发人员和测试人员头疼的软件性能 Bug，相信有过性能测试经历的读者都深有体会，发现、定位和修复性能 Bug 的成本非常高。

通过上述介绍，Bug 的衍生规律已经一目了然，那就是，**软件的 Bug 应该从它产生的那一刻起在最短的时间内被发现，否则，修复的成本会随着开发阶段的迁移而迅速增长。**

根据这种衍生规律，要在第一时间发现软件 Bug，就应该不断地去执行软件测试，这就是“多测”。

思考

为什么很多企业现在越来越重视单元测试？

2. Bug 分布模型 —— “测多”

在软件测试界里流转着一句谚语：customer is test guideline，中文意为客户行为是我们软件测试指南。其内涵意思是，凡是用户可能应用的场景，可能的操作行为，都应该是软件测试的内容。这句话说起来轻松，做起来就不那么容易了。因为从理论上来说，客户对软件系统的各种操作场景和行为仅受到想象力的限制，可以是五花八门，无任何原则上的边界。

比如一个邮件系统的用户可能突发奇想，创建上百个邮箱文件夹来存放其朋友的照片，也有可能过不了多久，此用户多次单击“删邮件”按钮来频繁删除邮件。那么，这些非正常的操作行为都是软件测试要考虑的内容吗？

通常我们做事的原则是把复杂变简单，而不是搞得更复杂。梳理一下软件系统的本质，我们会发现它有两个最基本的属性，首先是业务属性，即软件给用户提供的功能。还是那个邮件系统的例子，邮件系统的功能就是帮助实现收发邮件。其次是软件自身属性，即电子邮件不是用火车或汽车来运输，而是靠软件在虚拟网络上进行传递。既然是软件，那么就有可维护性、可扩展性、安全性等。

一般来说，所有的软件系统都有自己独特功能的业务属性，并且同时具有软件本质的共性，从图 1-2 中可以清晰地看出一个邮件系统所应该具有的功能。

在实际软件项目运作中，图 1-2 右边的业务属性往往由用户自己提出，并写在用户

需求规格说明书里，作为软件需求提交给软件企业；左边的软件属性对软件开发人员来说并不陌生，它们会在软件开发过程中得到考虑并实现，这其实是用户默认的、隐式的软件需求。

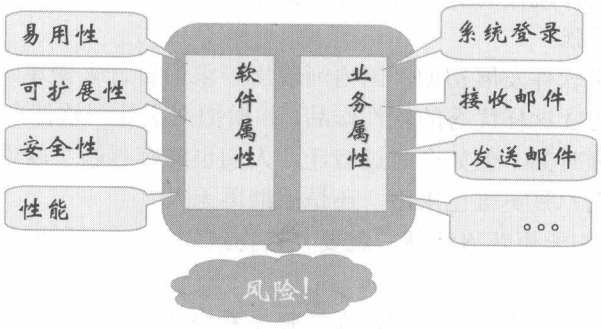

图 1-2　Bug 分布模型

如果认为上述问题已确定，就完事大吉那就错了，因为我们注意在图 1-2 中还有一部分灰色区域既不属于软件属性，也不是我们已知的用户需求，它的存在意味着总会有一部分用户的使用场景是软件设计人员和开发人员未曾考虑到或忽视的，这里往往隐藏着具有极强破坏性的软件 Bug。

一个优秀的软件测试团队的责任是，确保在已明确的软件属性和业务属性实现的基础上，多发现灰色区域的软件 Bug，从而达到提高软件质量的目的，最终提高客户的满意度。这就是“测多”。

案例

某电信服务提供商的主线产品是 ADSL，ADSL 最基本的功能有两个，第一是能够打电话，第二是能够上网。企业安排两个测试团队分别负责测试本产品的电话和上网功能，在临近产品发布给客户之时，有个测试人员突发奇想，认为如果用户在用 ADSL 上网的同时又试图拨打电话会怎样，这是以前两个团队都没有测试的一个场景。结果他发现在这个操作场景下，ADSL 软件会重启，这是一个非常严重的 Bug，所幸的是这个 Bug 在产品发布之前被及时发现并解决。

思考

结合“customer is test guideline”，想想从软件测试角度来看，软件项目和软件产品有什么异同？

“多测”和“测多”两种策略已使得软件测试的工作面临巨大的压力和挑战，“破屋又逢连阴雨”，我们又遇到了另外一个困境。这就是下面即将谈到的软件行业的另一个趋势——软件规模的迅速扩大对软件测试成本的要求。

1.1.2 软件成本困境

成本困境体现在软件规模的扩张对软件测试提出了更高的成本投入要求。

1. 测试规模的扩张

十几年前的计算机软件发展初期，国内涌现出了求伯君、朱崇军等软件高手，凭一人之力即可写出 WPS、CCED 这样优秀的软件作品，并横扫文字处理软件市场，对软件人员来说，这无疑是最辉煌的顶峰时刻。但如今时过境迁，人们也鲜有再闻“软件枭雄”，“软件大师”的说法。这是为什么呢？是英雄已末路，还是后继再无人？

《大学》中写到人们要想获得智慧，就要“格物致知”，下面我们就一点一点地“格物”，寻找英雄失落的原因。

- 首先不得不承认一点，以人们现在的眼光看来，当年大师们的软件作品代码行规模并不大，据说 WPS 卖得最火的第一版总共几千行的源代码，这个数量级也就是现在一个 Java 字符集转换函数的源代码实现的规模。因此，软件规模不大是当时软件人员能够独力完成作品的一个必要条件。
- 面向对象技术的产生和发展为代码复用提供了技术保障，并且在客观上极大地促进了软件规模的迅速膨胀。Java 是面向对象的代表，企业和个人选择 Java 的一个重要因素是 Java 的类库和开源非常之多，用户想用的功能，几乎都能在 Internet 上找到，使用起来也十分简单。这就是面向对象给我们带来的好处，但同时也增大了软件的规模。
- 计算机网络硬件及软件的发展使得分布式软件架构的可行性变得现实，使得软件系统的应用范围和场景进一步扩大。软件系统越来越重视交互和协作，如多个模块服务的交叉调用，网间的交互安全等。因此，软件系统从单机版到 C/S 架构，又到 B/S 架构，又到三层架构和四层架构，软件系统的架构也在日趋复杂和多层次化，一个人是无法独力设计并开发这样一个庞大的系统的，所以，软件开发组织模式也从单枪匹马到手工作坊，又到大规模合作化开发。这是一个不可逆转的趋势。

从上面的分析可以看出，当今不是英雄末路，也不是后继无人，而是软件的“时势”变了，在新的“时势”下，软件规模在不可逆转地迅速扩大，需要更多的人共同协作开发。那么，软件规模的扩大给软件测试带来什么影响呢？

2. 测试成本的考虑

面向对象技术的发展使得开发人员越来越容易在短的时间内写出一个界面美观、功能完善的软件。比如，使用 Eclipse，Jdeveloper 等 IDE 工具，调用现成的界面组件和应用服务容器，几天之内就能开发完成一个简单的 Web 网站。但这对测试人员来说却是一种挑战，因为，他们不仅要测试开发人员写的那部分代码，而且还要确保所有的代码（包括复用的代码）都集成在一起并正确无误地工作，他们要比以前付出更多的工作量，来设计更多的测试案例，执行更多的测试场景。这实质上也就是提高了测试的成本。

（1）人力成本。

人是任何工作中最重要的资源。对于软件测试来说，从测试案例的设计，到执行，再到 Bug 的跟踪，都需要软件测试人员的参与。一般来说，在一个比较成熟的软件测试组织里，软件测试案例的数量和测试人员的数量有一种稳定的对应关系，软件测试案例越多，人员数量也越多。

案例

某知名软件外企对于软件测试人力成本的估算有一种经验的参考数据，平均一名测试人员负责 150 个案例。也就是说，如果一个系统测试设计出来的测试案例是 600 个，那么，就可以大致估算这个项目需要 4 名测试人员。

结合上面所说，面向对象技术使得软件规模得以扩大，然后导致软件测试案例的数量增多，自然而然地，软件测试的人力投入就要增多，否则就很难确保软件的质量，这是一个传导链条。

（2）时间成本。

软件测试这部巨大的“机器”一旦被前期的软件测试案例驱动，就进入自循环调节的回归测试的周期律，即发现 Bug→回归测试→再发现 Bug→再回归测试。可以看到，在这个周期里，有至少一半的时间花费在测试执行上。假如一个测试案例手工执行需要花费 10min 时间进行计算，那么 100 个测试案例就是 1000min，即 1000/60=16.6h，按理想的 8h 工作日计算，那么每一轮的回归测试，仅执行就要花费整整两个工作日，这个时间的耗费是巨大的。

案例

某知名公司在其产品线上有一个插件模块，这个插件是安装在 Micsrosoft 的 Outlook 上，提供一些常用功能，比如收发 Mail、Calendar 创建等。而此公司的测试部门仅为这个插件就设计了 6000 多个 test case（想象一下测试场景组合，插件的安装和运行在不同的 Outlook+SP 版本，不同的 Windows+SP 版本），如图 1-3 所示。

操作系统	5	Win98,Win2003,Win2000,Windows XP,Visita,
测试数据	10	10种字符串
测试语言	4	English, French, German, Japanese
测试案例	5	登录,发邮件，收邮件,删除邮件，邮箱管理
测试执行案例总计	**1 X 5 X 10 X 4 X 5　= 1.000**	

图 1-3　Outlook 插件产品测试场景组合

这个数目是如此巨大，使得测试执行和产品周期产生了巨大的矛盾。这种矛盾体现在，当每个新版本发布时，如果做一遍完整地测试，一个人手工测试执行一遍 6000 多个 test case 就要半个月的时间，而产品版本的构建周期也就一周左右的时间，也就是说连测试执行一遍的时间都不够，更别说 Bug 的提交和跟踪了，测试的速度远远跟不上产品的发布速度。

在软件测试的实际工作中，人力和时间成本往往都是受限制的，当人力和时间都不够时，怎么办？只能采取拆东墙补西墙的办法，以人力换时间，或以时间换人力。因此，在人力资源不足的测试组织里，我们经常看到软件测试人员成了救火队员，哪里有急情，就冲向哪里。从长远的眼光来看，这其实对测试队伍的培养是很不利的。

由以上分析可以看到，软件行业的质量困境和成本困境对软件测试工作提出了严峻的挑战和更高的甚至苛刻的要求。那么，作为软件测试人员，我们有什么办法来应对这种问题呢？有，即软件自动化测试。

1.2 “时势造英雄”——软件测试自动化的异军突起

要确保质量，就要对软件进行“多测”和“测多”，这就意味着要增加测试人力资源的投入和成本投入，这是一种两难选择，要么保质量，要么保成本，两者如熊掌和鱼不可兼得。

1.2.1 软件自动化测试如何代替手工测试

软件自动化测试（test automation）指的是以程序运行的方法替代人工测试，从而达到减少手工工作量，提高测试效率的目的。

自动化测试能替代手工测试的所有工作吗？它是怎样实现替代的呢？下面一一进行分析。

一般地，手工执行一个案例的时候，总会关心以下 3 个要素。

（1）测试环境。

（2）测试执行步骤。

（3）测试预期结果。

可以通过以下 3 个问题来更进一步地理解上述 3 个要素。