Effective Software Test Automation

国外IT精品丛书

高效软件测试自动化

开发软件测试自动化工具

[美] Kanglin Li
Mengqi Wu 著

曹文静 谈利群 等译

SYBEX

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Effective Software Test Automation

高效软件测试自动化

[美] Kanglin Li
Mengqi Wu 著

曹文静 谈利群 等译

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内 容 简 介

本书逐步引导你学习一种可重用的、适合任何开发环境的自动化测试工具的创建方法，同时，还为你合理部署工程提供了专家级的指导。本书前两章简述了软件测试技术和当前测试工具存在的缺点，并提出了避免这些缺点需要进行的工作。接着进入对自动化测试工具所用的.NET 编程技术的讨论，然后，使用这些技术完成了能够自动生成测试脚本以进行单元测试、集成测试以及回归测试的自动化测试工具的开发。在对.NET 的介绍中，主要涉及了命名空间、类、Reflection、CodeDom 以及与 Excel，XML 相结合的应用技术，并提供了大量的开发例程，供读者学习和练习。

本书适用于具有一定软件开发和测试经验的程序员、测试人员、开发人员，以及软件项目的管理者。

版权贸易合同登记号　图字：01-2004-1359

图书在版编目(CIP)数据

高效软件测试自动化/(美)李(Li，K.)等著；曹文静等译. —北京：电子工业出版社，2004.8

书名原文：Effective Software Test Automation

ISBN 7-121-00128-4

Ⅰ.高…　Ⅱ.①李…②曹…　Ⅲ.软件—测试—自动化　Ⅳ.TP311.5

中国版本图书馆 CIP 数据核字(2004)第 070023 号

责任编辑：徐云鹏

特约编辑：卢国俊

印　　刷：北京天竺颖华印刷厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱　邮编：100036

北京市海淀区翠微东里甲 2 号　邮编：100036

经　　销：各地新华书店

开　　本：787×1092　1/16　印张：23.375　字数：590 千字

印　　次：2004 年 8 月第 1 次印刷

定　　价：37.00 元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系。联系电话：(010)-68279077。质量投诉请发邮件 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

致　谢

我希望这本书对你来说能成为一本珍贵的软件开发参考书，特别是针对软件测试自动化，其目标是减少错误及成本，同时提高软件产品的质量和盈利性。

我第一个要谢的是购置和发展编辑 Tom Cirtin，是他让我能够写出这本书。他总是努力工作，充满智慧、热情和灵感。他总是最棒的。同时我还要谢谢 Jordan Gold。

Sybex 小组的其他成员还包括 Acey J. Bunch，Judy Flynn，Erica Yee 等。他们完成了大量的技术编辑、版面和制作编辑工作，并且提供了许多建议和评价，我想为他们优秀、勤恳的工作表达我的谢意。

我还想向我以前在安捷伦的同事 Susan Powell，Kenneth Ward，Kevin Keirn，Chuck Heller，Pat Kennedy，Greg Kuziej，Phil Driggers，Jun Liu，Ting Wu，Jing Dong Zhang，Dave Willis，Alicia，Jerry Metz，Mike Hawes 和 Jayson Jackman 表达谢意，我们共度了美好的时光，还一起出色地完成了许多工作。同时，我还要谢谢 Marcie 和 Bob Hervey，Steve 和 Molly Child 以及他们家庭对我的友善。

我要特别感谢 Ray Gonzalas 先生。他是国家证券机构的一名软件工程师。他是最早的一字一句阅读此书原稿的人之一，并为我提供了宝贵的意见和具有建设性的建议，指出许多需要修改之处。这本书最近一版体现出了他的巨大努力。同样我还要特别感谢 Lena Berrios，作为另一位最早阅读此书原稿的读者，她做了许多修改工作使语言更加优美、写作风格更加条理。我曾担心，作为一个专业的图书管理者和语言学家，她也许会觉得阅读一本计算机书籍很无趣。幸运的是，她后来告诉我她从中学到了不少东西，并鼓励我完成这个项目。

我要诚挚地感谢我的硕士学位导师，北卡罗莱纳州立大学的 Aziz Amoozegar 博士和北卡罗莱纳州立航空科技大学的 M. R. Reddy 博士，是他们教会我如何去思考、去写作。

最后我还要表达我对来自国家标准与科技协会的 Gregory Tassey 博士的深深谢意，他慷慨地允许我使用他已发表的著作中的事实和数据。

引　言

目前有许多软件测试管理方面的书籍，它们对于软件测试自动化的讨论，局限在介绍第三方测试工具。本书描述了开发一个完全自动化软件测试工具所用到的技术，这样的工具能够为连续的单元测试、集成测试和回归测试产生测试脚本。

普遍存在的软件缺陷导致了一次又一次的经济损失。分析和测试软件活动不再被看做孤立存在的实体，而是作为软件开发过程的一个单元存在，现在的软件开发团体为分析和测试软件过程投入了更多的时间和资源。编写测试代码的代价并不亚于产品开发本身，为了以尽可能小的代价保证软件质量，软件开发组织鼓励开发者和测试者达到下面的这些目标：

- 更快更精确地定位软件中的缺陷位置
- 在软件开发生命周期中更早地检测出缺陷
- 在产品发布前，尽可能减少软件中的缺陷

使用改良的测试工具，能够减少软件开发代价，提高软件质量。一个自动测试工具必须具有如下特性：

- 具有精确性、正确性、可靠性、互用性和依从性
- 具有友好的界面，易于学习和操作
- 增强的容错和自动错误恢复能力
- 具有高性能高效率算法
- 具有稳定成熟的最后工具产品，以后可以维护和升级
- 具有安装和卸载方面的可移植性，具有适应性和安全性

许多商业软件测试工具的开发商声称，他们的产品有能力管理各种类型的软件测试，能够满足用户的需求，但实际上这些工具都有局限性。例如，有些工具要求用户记录一系列的鼠标和键盘动作，有些工具要求用户用特定的脚本语言编写测试脚本，或工具仅为模块的某个函数（成员）自动产生一个测试脚本。此外，这些工具和方法产生的测试脚本，经过编辑和调试，然后才能用于测试。这些工具不能自动产生测试数据。利用这些工具进行集成测试，需要手工编写大量的桩模块，编写桩模块的过程还需要猜测桩模块所模仿的真实模块的特性。

人们期望可以买到一个完全自动化的软件测试工具，期望这样的测试工具能够自动实现整个测试任务，包括从产生测试脚本和编写测试用例，到表达测试结果和修改缺陷的整个测试过程。但测试工具开发商却跟不上软件项目不断增加的复杂性，和项目开发采用技术的先进性。另外，软件产品可能包含开发商的商业秘密，商业测试工具没有能力进行测试，人们经常不得不自己开发工具来弥补这些缺憾。本书提出了一种开发和升级完全自动化测试工具的方法。

测试工具开发商能罗列出工具的数百个功能，测试者确实喜欢使用这些功能，而且这些功能对改善软件质量是很重要的。但是编辑和调试产生的测试脚本的过程乏味而耗时，经常妨碍彻底的软件测试。结果往往是，当软件产品中还有严重的错误时，就提交给最终用

户，用于支持开发、生产、售出、售后服务过程。事实上，所有使用这样软件产品的部门，都可能承担软件失效导致的损失。为了弥补当前不充分的测试工作，本书介绍了一种自动化的测试方法，这种方法使得数据编辑步骤最小化，自动产生用于测试整个应用程序的测试脚本，而不需要手工编辑和调试测试脚本，仅需简单地把被测的应用程序提交给最后的工具产品，就自动输出测试结果。

本书的读者对象

在对当前工具和支撑技术基础的长期使用过程中，一些人成功了，更多的人遭受了挫折。自动化程度不够高，测试效率低，测试脚本和数据产生方法需要改善，这些都是软件测试自动化中存在的问题。专家对这些问题的解决方案是自行开发测试工具，而不是购买商业工具，这是因为用于商业工具开发的支撑技术基础是不充分的。本书的读者对象是参与软件工程过程和软件测试自动化过程的人员。通过本书介绍的方法，可以深入理解当前测试工具有限的自动化能力，学习如何改善当前的支撑技术基础，从而实现完全自动化的软件测试过程。

本书读者对象是：有意于用更有效的方法实现软件测试的软件工程师。本书介绍的AutomatedTest 工具可以作为一个独立的软件测试工具存在，也可以嵌入商业工具，作为一个附属组件存在。

本书假设读者对软件开发和测试有一定的经验。对于中级到高级水平的程序员，本书的解说和示例很容易理解和实现。对于软件测试工程师，掌握软件测试的基本原理知识是至关重要的，而只有掌握编程和测试两方面的知识，才能解决软件测试自动化的问题。本书的内容囊括了用 C# 示例的可靠编程技术，然后，从技术逐渐过渡到开发一个完全自动化的测试工具。虽然示例代码是在 Microsoft Windows 平台上以 C# 语言编写，但其思想可以应用到其他平台和语言上。

据报导，每年美国都有巨大的经济损失是由软件失效引起的，其中，软件开发商承担了大约一半的经济损失，因此，开发组织的高级管理人员最感兴趣的是改善软件测试方法。软件最终用户承担了另一半的经济损失，因此，对于最终用户来说，成立维护软件小组，了解测试方法会有助于高效率地应用软件。

本书主要内容

为了顺利阐明思想，前两章介绍软件测试技术和编程语言，第 2 章还罗列了当前测试工具的优缺点，详细说明了克服缺点的目标。第 3 章到第 6 章讨论用于 AutomatedTest 工具开发的. NET 编程。在基本原理的讨论中，只要时机成熟，就把新的技术应用到 AutomatedTest 工具的开发中。在了解了必需的知识后，第 7 章到第 11 章使用第 3 章到第 6 章介绍的编程技术，继续完成 AutomatedTest 工具的开发，最后的 AutomatedTest 工具能够自动产生测试脚本，从而进行单元测试、集成测试和回归测试，还对揭示的缺陷进行报告。最后，第 12 章通过一些实际软件的测试示例，对本书进行了总结。

本书的组织如下：

第 1 章“软件测试概述”，描述了一些用于动态测试软件的. NET 内建技术，并举例说明了如何把这些技术集成到自动软件测试过程中。

第 2 章“当前测试的支撑技术基础和本书提出的测试方法”，简要回顾了一些商业自动测试工具，阐述了本书提出的测试方法。研究表明，现有的工具对于软件测试是不够的，本章的目的是证明新的测试方法的能力。新的测试方法不仅为测试工具增加了更多的测试功能，也能实现对新开发的复杂软件项目的完全自动化测试。现有的工具在各种环境中测试软件（例如，Java，UNIX，Linux 和 Windows），因此，这里介绍的方法不限于微软平台和产品，而是可以扩展到其他的开发环境。

第 3 章“. NET 命名空间及类在软件测试中的应用”，初步介绍了自动测试任务。

第 4 章“. NET Reflection 在测试自动化中的应用”，介绍了. NET Reflection 命名空间。就像棱镜分解太阳光一样，Reflection 命名空间可以映像被测软件。

第 5 章“电子数据表和 XML 在测试数据存储中的应用”，介绍了一种编码时把测试用例存储到 XML 文档中的方法。这一章还讨论了在测试脚本中如何对 MS Excel 电子数据表编程，用于存储测试信息和描述测试结果。

第 6 章“. NET CodeDom 命名空间”，描述了基于 Reflection 命名空间揭示的信息，以及使用. NET CodeDom 命名空间产生测试脚本的方法。

第 7 章“产生测试脚本”，基于类、方法和参数，实现使用. NET CodeDom 动态编写测试脚本的方法，这样产生的一个测试脚本能够测试程序集合的所有成员。为程序集合产生一个测试脚本的优点是：执行的测试脚本能返回被测类的一个完整对象。返回的对象可重用于集成测试或对象传递参数方法测试。

第 8 章“集成测试”，研究了一种自动测试对象传递参数的方法，用于集成测试。为了自动集成测试，本书没有利用现有的搭建桩模块方法或模拟对象方法，而是介绍一种自底向上的集成测试方法，可以重用以前产生的测试脚本进行自动集成测试。

第 9 章“验证、确认、描述”，描述了自动测试验证、确认和结果描述的过程。这一章还介绍了一种方法，这种方法描述了在最早的设计阶段，如何使用工具测试仍在开发中的软件。验证、确认以及揭示错误的结果都在电子数据表中进行了描述。

第 10 章“完成 AutomatedTest 工具”，完成了测试工具的开发，阐述了如何改善图形用户界面。

第 11 章“增加 AutomatedTest 工具的 Windows 注册表测试功能”，介绍了如何升级 AutomatedTest 工具，用于 Windows 系统注册表测试。读者可以为测试工具增加更多的测试功能，从而满足更多特定的需求。

第 12 章“测试 AutomatedTest 工具”，通过使用 AutomatedTest 工具测试真实的软件，对本书进行了总结。这一章还阐述了如何使用本书介绍的方法，测试 AutomatedTest 工具本身。

本书的示例

开始的一些例子用于说明 C # 的编程规则，目的是使用编程语言预定义的方法实现 AutomatedTest 工具。本书共有三类示例代码：

- 演示 C# 编程的简单示例
- AutomatedTest 工具测试的示例工程
- AutomatedTest 工具本身的示例代码

大部分第一类代码示例在第 3 章到第 6 章中出现。之后，第 7 章到第 11 章讲解测试工程的自动化，其中的示例代码属于第三类。第二类示例代码仅有三个，模拟一个真实的被测程序集合，这三个示例分别在第 4 章、第 8 章和第 11 章中实现，这些代码示例出现在每次 AutomatedTest 工具编码前，在每一章的结尾，把示例提交给新编码的测试工具。

本书中，示例是托管程序集合，第 5 章简要介绍了非托管和托管程序集合间的互用性。当需要测试一个非托管 COM 组件时，可以把它简单转化成一个托管程序集合，就可以用 AutomatedTest 工具测试了。

从第 3 章开始，每一章都为工具增加一些代码，在每一章的最后，编译示例代码，产生一个可执行程序集合，从而使测试工具达到一个更高的自动化水平，直到本书的最后，实现了测试工具的完全自动化。

第 4 章的代码的加入，使 AutomatedTest 工具具有了映射被测程序集合的功能。第 5 章实现 XML 文档和 MS Excel 电子数据表，用于测试用例数据存储。第 6 章介绍了 CodeDom 的许多基本原理，用于自动产生测试脚本，示例程序示范了如何使用 CodeDom 产生程序，但这一章中没有把代码加入 AutomatedTest 工具。

第 7 章继续编码 AutomatedTest 工具，使工具具有为给定的集合产生测试脚本的功能。第 8 章代码的加入，使工具具有了测试对象传递参数的能力，从而进行集成测试。第 9 章讨论了关于验证、确认和结果描述方面的重要测试话题，这一章之后，工具可以产生一个能测试全部成员的测试脚本，然而，还需要人工输入一条命令，运行和配置测试脚本。第 10 章为工具加入最后一批代码（用于编译和配置测试脚本程序集合），从而得到一个完全自动化的测试工具。这时，只需把程序集合提交给工具，就可以获得程序集合中被测类、方法和属性的测试结果。最后，第 11 章讨论了如何升级工具，使它具有更多的功能，并示例说明了如何增加方法，从而产生测试 Windows 注册表的代码。

下载本书源代码的方法

访问网站 www. sybex. com，使用题目、作者或 ISBN 号 4320 搜索到本书，就可以下载每一章的示例和工具代码了，还可下载完整编译版本的工具，从而直接应用到软件项目实践中。

把 AutomatedTest. exe 文件复制到计算机系统中，然后运行。AutomatedTest. exe 对系统的最小要求如下：

- Windows 95/98/2000/NT/XP
- 预安装的. NET 框架(framework)
- 预安装的 MS Excel
- 20MB 可用硬盘空间

本书的示例代码是在 Visual Studio . NET 2003 集成开发环境（IDE）中开发的，当然，也可以免费下载其他的开放. NET 集成开发环境。

Eclipsing . NET

IBM 的 Eclipse . NET 是开放源码产品，它的工作平台是 Windows XP/2000/NT/98/95。首先到网站 www. eclipse. org 下载 Eclipse . NET 的组件，然后到网站 http://msdn. microsoft. com/netframework/technologyinfo/howtoget/default. aspx 免费下载 Microsoft . NET SDK，下载 eclipse-SDK-2. 1. 2-win32. zip 之后，和 Microsoft . NET SDK 一起安装，然后获得开放源码 C＃插件程序，安装到 Eclipse . NET IDE 中。（更多的下载和安装这些程序的详细信息，参看网站 www. sys-con. com/webservices/articleprint. cfm? id=360. ）

＃develop（SharpDevelop 的缩写）

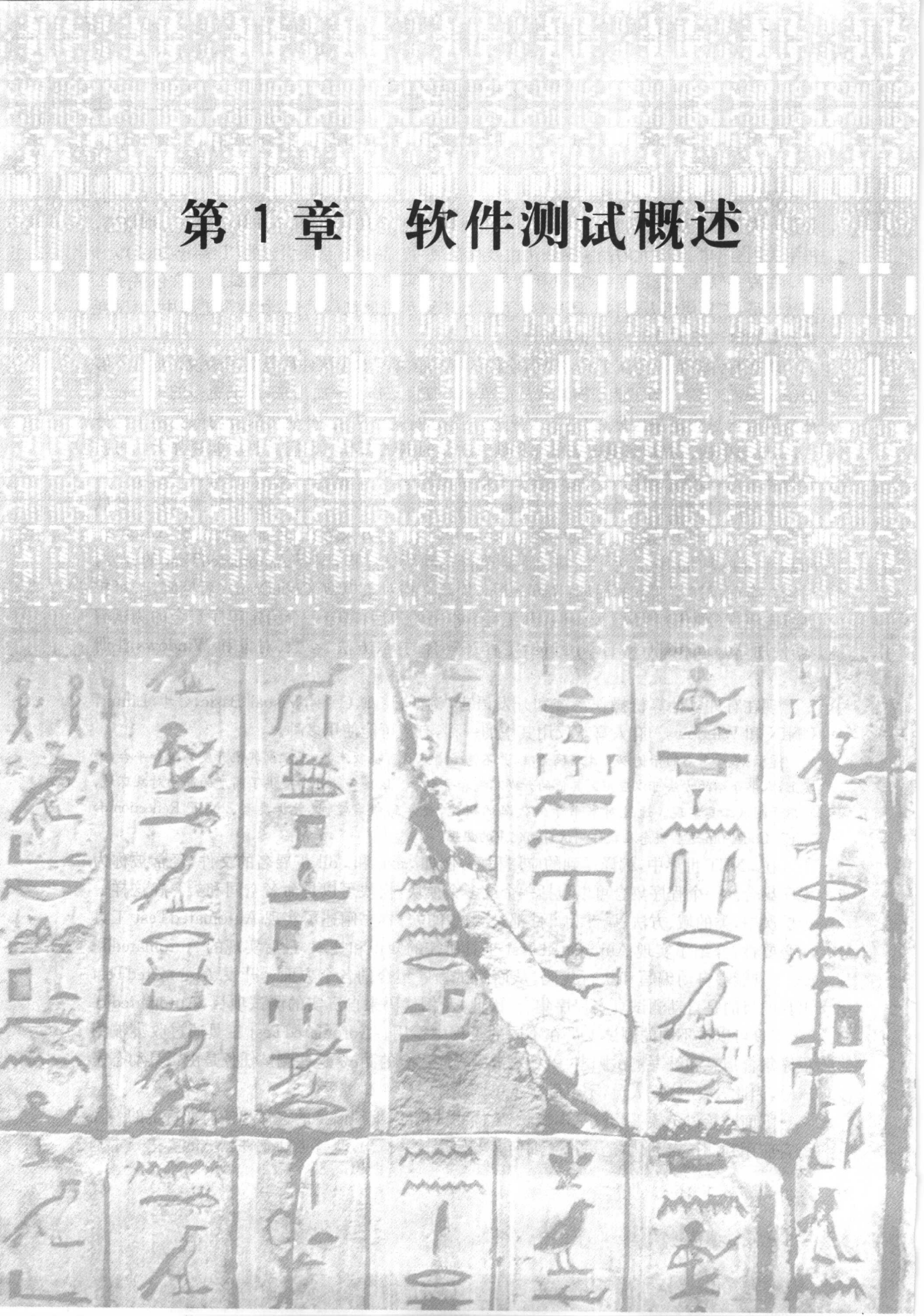
这是另外一个微软. NET 平台上的开放源码集成开发环境（IDE），用于开发 C＃和 VB. NET 工程，可以到网站 www. icsharpcode. net/OpenSource/SD/Default. aspx 下载＃develop。

DotGNU Portable . NET

这个开放源码工具包括 C＃编译器、汇编程序和运行时引擎。该工具最初的工作平台是 GNU/Linux，后来扩展到 Windows，Solaris，NetBSD，FreeBSD 和 MacOS X 平台。可以到网站 www. southern-storm. com. au/portable_net. html 下载这个工具。

目　录

第 1 章　软件测试概述



使用当前的测试技术，测试后的软件产品中都会残存错误。软件编码时能检测和修正一部分错误，模块集成到系统时的形式化测试中，也能检测并修正一些错误，然而，软件中仍然残存错误，其中的一些错误不得不留到以后修正(Beizer 1990)。测试是软件产品成功的必要先决条件，但当前的测试技术本身具有难以实现、单调、耗时和不充分的缺点。据报道，每年美国有 595 亿美元的经济损失是由残存在最后产品中的错误引起的(Tassey 2002)。

在许多软件开发工程中，测试耗费占工程总预算的 25%～50%。测试组通常包括手工测试人员、工具使用人员和工具开发人员。由于要尽可能彻底测试，在整个工程中，测试耗费预算和测试组人员都占有重要的地位。

现在有许多商业测试工具，然而，它们的测试能力相对于软件质量需求来说是远远不够的(Tassey 2002)。为了能把测试者从简单的重复性工作中解放出来，而把更多的精力投入在高风险区域中解决问题，这些工具实现了部分测试自动化，自动化部分局限在两个任务上：实现简单逆工程和产生记录鼠标和键盘动作的测试脚本。人们需要功能更强大的可扩展测试工具，这样的测试工具能提供更多自动化特性，从而能跟上快速发展的软件技术。

本书的目的是开发 AutomatedTest 测试工具，以最少的人工干预，彻底测试复杂的软件产品。从本书中，读者不仅能学到 AutomatedTest 的开发方法，还能学到实现 AutomatedTest 所必需的知识。为了控制和逐渐改善自动测试过程，围绕 AutomatedTest 的开发，讨论并全面描述了如下的话题：Reflection、代码文档对象模型(CodeDom)、后期绑定、可扩展标志语言(XML)和 MS Excel API 编程。本书的最后，基于一个给定程序集合的测试需求，使用 AutomatedTest 自动测试给定程序集合的类、方法、参数、对象和 Windows 注册表。

现在有许多计算机操作系统和开发语言，例如：Java，C++，Visual Basic，C#，Linux，UNIX 和 Windows。有人喜欢仅用其中的一种，有人可能使用多种。

提示：尽管本书的示例以 C# 编码为准，并不意味着只认可微软产品，或挑剔其他开发环境和平台，事实上，本书介绍的方法可以应用到其他的开发环境和平台中。Java 和 .NET 提供了高度的面向对象环境，可用于测试工具实现。通过对本书示例代码的研究，会学到命名空间、数据类型、.NET Reflection 和 .NET CodeDom 这些概念在软件测试自动化中的应用。

在 .NET 世界中，编译得到的应用程序，例如 .exe 和 .dll 扩展名的文件，经常被称为程序集合。一个程序集合可以包括一个或多个模块，模块可以是命名空间和类。在这样一种层次中，类的域、方法、属性和事件都是测试单位。本书阐述了实现 AutomatedTest 工具的必要性，介绍了实现 AutomatedTest 工具所需的知识和技术，最后实现的 AutomatedTest 工具能够自动编写和执行代码，从而测试程序集合的各个方面。开发 AutomatedTest 工具的目的是自动测试真实程序集合，因此，示例代码来自真实的测试项目，AutomatedTest 工具可以被不同的测试人员在不同的工程中重用。AutomatedTest 工具通过收集被测程序集合信息和编写测试脚本，引导对被测程序集合的测试，这些测试任务是由工具动态完成的，因此，对于测试人员，不再困扰于单调或耗时的测试过程。

下面开始讨论软件测试基本概念、软件测试目的、编程语言的选择和如何把特定的编程语言应用到测试自动化，还讨论了如何有效使用这些概念，改善当前软件测试的支撑技术基础。

1.1　软件测试的目的

现在美国每年软件的销售额达到 1 800 亿美元，而由软件中残存缺陷引起的灾难时常发生。

1999 年 4 月，软件缺陷导致一颗价值 12 亿美元的卫星在卡纳维拉尔角基地（Cape Canaveral）发射失败，这可能是软件史上造成损失最大的一次软件失效，这次事件引发了军民两方对美国空间发射程序的彻底审查，包括软件集成和测试过程。

可能有人还记得 1999 年 10 月与地面失去联系的 NASA（美国国家航空和宇宙宇航局）的火星气候探测器，那次事件是由英制单位转换成公制单位部分软件失效引起的。2002 年，我为视力测试仪器开发了一个模块，用于确认没有度量单位混合。若为火星气候探测卫星开发过这样的模块，那么今天，这颗卫星应该还运行在轨道中。

本书的读者对象是每个对软件质量感兴趣的人，包括高级管理人员、软件项目管理人员、软件开发人员、软件测试人员和软件最终用户。当启动一个新项目时，软件团体都遵守开发模型，软件最终用户、开发人员、测试人员和高层管理人员都参与对需求、规格和测试策略的定义。在大多数模型中，在开发过程伊始就制定测试计划，并和开发生存周期并行发展。只有在理解的基础上才能测试产品，对于测试人员，测试新开发的产品的过程也是学习的过程，因此，这个过程所耗费的时间和精力取决于产品复杂性和测试人员的经验。在很大程度上，使用自动工具能防止测试人员在学习新软件上耗费过多时间，从而能够把更多的精力投入到更复杂和高风险的问题上。

软件测试是执行应用程序，检测缺陷，检验应用程序是否满足指定的需求的过程。在软件开发生存周期中，开发人员和测试人员协同工作，检测缺陷，确保产品质量。提交给用户的软件产品必须包括所有需要的功能，并与用户的硬件兼容。

长期以来，都是手工进行软件测试，即测试人员按预定义的过程运行应用程序。自从软件业兴起以来，人们为自动化软件测试过程做了很多工作，许多公司开发商业软件测试工具，用于检测缺陷，在产品发行前修正缺陷。正如前面提到的，这些工具在某些方面具有自动化的功能（例如实现逆工程和编写测试脚本），但经常也具有如下的缺点：

- 测试脚本常常需要调试
- 没有一个测试工具能够独立完成整个测试过程
- 这些工具实现的测试过程可能和软件设计过程不一致
- 逆工程过程和测试脚本产生过程是完全分开的两个过程

使用工具为程序集合的每个成员产生或记录一个测试脚本，对测试人员来说，常常是一项任务繁重的工作；利用工具编辑和存档测试数据纯粹是手工完成的。因此，这些工具的自动化能力是有限的。

使用本书开发的自动软件测试工具，测试人员不必手工编写测试脚本或记录测试场景，而且使得软件测试过程中的人工干预最小化。该工具设计为可重用的，能够满足大部分软件产品的测试需求。换句话说，读者将学习如何构造测试其他代码的可重用模块。