Effective GUI Test Automation

图形用户界面测试自动化

[美] Kanglin Li
Mengqi Wu 著

王轶昆 等译

开发一个自动化的GUI测试工具

SYBEX

美迪亚

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

Effective GUI Test Automation

图形用户界面测试自动化

〔美〕 Kanglin Li
Mengqi Wu 著

王轶昆 等译

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

内 容 简 介

目前市场上可见的测试工具尽管可以一定程度地满足用户对软件测试的不同需求，但是在某些方面仍有很大的局限性，例如这些工具所使用的捕获/回放技术要求用户手工进行测试脚本的记录工作，或者使用特殊的脚本语言编写测试脚本等。本书所提出的全新的测试GUI的方法，可以使测试者无须手工编写测试脚本，无需繁琐而且枯燥的捕获/回放工作，实现最大化的自动化测试：用户只需输入一个待测的软件，就可以自动输出测试报告。本书提供的丰富的代码示例程序不仅可以帮助读者理解测试的概念，更可以直接在工程中应用，提供有价值的帮助。

版权贸易合同登记号　图字：01-2004-6703

图书在版编目（CIP）数据

图形用户界面测试自动化/（美）李（Li，K.）著；王轶昆等译.—北京：电子工业出版社，2005.4
书名原文：Effective GUI Test Automation
ISBN 7-121-01000-3

Ⅰ. 图…　Ⅱ. ①李… ②王…　Ⅲ. 软件—测试　Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2005）第016487号

责任编辑：陈　宇
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：600千字
印　　次：2005年4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。质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。

献给Li Xuanhua和Tang Xunwei

纪念Luo Youai、Luo Xulin和Li Congyang

致　谢

感谢所有Sybex同仁，尤其是Tom Cirtin，他使这本书得以出版；感谢技术专家Acey J. Bunch，以及Erica Yee和Judy Flynn为本书做出的贡献。

我想感谢Sybex总裁Rodnay Zaks，他和我签订了出版合同。我也感谢Sybex团队中其他的人。

同时，我还要感谢其他的人，尤其是我前一本书的读者，他们向我提供他们的意见和建议。在编写这本书的过程中，第3章中的测试猴子经常令我想起我的早年学生生涯。无数夏日的晚上，我和我的朋友们都会听村里的老人给我们讲述猴王的故事，我从这些老人和我的朋友们那里学到的知识，令我今天能够写出这本书。

在我还是个小孩子的时候，我的叔叔OuYang Minghong把我放在一个竹篮里去看望我的奶奶。我的叔叔Zuomei和Zuodian则会把我放在独轮车里或者背着我去看电影，要么就去抓鱼。从我五岁开始，我的姑姑Youai把我带到了学校，教我如何用手指头数数。

我八九岁的时候，尚不会游泳，有一次掉进池塘，是Meiqing冒着生命危险，和我的叔叔Zuohuang、Guicheng和另外一个我忘记名字的叔叔一起把我从水里救了出来。Meiqing一直像我的大哥哥一样。后来在我20岁的时候，我学会了游泳。在一个夏日我横渡了湘江，后来又在一个寒冷的秋天下午横渡了涞水河。感谢所有教我游泳的人们：ZengXilong、Li Wenbing、Kuang Chuanglun、Bai Maoye、Chen Xiaohu、Zeng Yigui、He Hechu、Wen Xiaoping、Long Yongcheng和Xie Hengbing。

Tang Duzhang、Li Zuojin、Li Zuojun、Luo Xixing、Chen Xinhua和Kuang Chuangren，我和你们一起度过了最美好的中学时代。那时我们知道了食物是多么地可贵。

我们还一起发现男孩子很害羞，而女孩子则富有同情心。感谢所有我以前的同学们，你们组成了我童年的美好回忆。

译 者 的 话

作为一个测试工程师，在使用商用的测试工具时，一定会对商用工具在性能上的不充分性有深刻的体会。目前市场上的测试工具，绝大多数使用捕获/回放技术记录测试脚本，使用者必须编辑和调试这些测试脚本，并加入验证点。这样的过程很难被称为“自动化测试”，更不要说记录脚本是多么繁重而且重复的过程了。

本书可以帮助摆脱这样的困境。使用这本书中提出的概念和技术，可以自己生成完全自动化的GUI测试工具，这是真正的“自动化”：只要向工具中输入一个应用，就可以等待工具生成测试报告。测试过程中需要的测试数据和验证数据还可以单独存储，因此，为一个项目生成的工具，同样可以在其他的项目中重用，大大节约了生产力资源。

本书的一大特色是其中丰富的代码例子，而且全部是可以直接使用的。如果不想手工输入，亦可以从网站直接下载。甚至可以先下载一个可执行的版本，一边学习，一边使用，同时增强工具的性能。

相信这本好书可以为程序员、测试工程师以及项目经理提供更加平滑有效的测试手段，从而使最终的软件产品尽可能无暇地发布。

简　介

目前有很多关于软件测试管理的书籍。当涉及软件测试自动化时，这些书籍会介绍第三方的测试工具。我曾经使用过很多商用的软件测试工具。它们的开发者声称，这些工具可以管理不同类型的软件测试，并且满足企业的要求。但是它们都存在缺陷。例如，很多GUI测试工具要求用户记录一系列的鼠标点击和键盘击键。其他的工具会要求用户使用特定的测试脚本语言编写测试脚本。更有甚者，在完成所需的测试之前，用户需要编辑和调试利用这些工具和方法生成的脚本。

本书提出了自动进行图形用户接口（GUI）测试的概念。本书中的例代码可以成为完整的自动化GUI测试工具的基础。使用这个工具，用户无需记录、编辑和调试测试脚本。测试者可以把时间更多地用在生成测试用例和测试运行上。

本书的读者

一直以来，软件工程师都倚赖于当前测试工具厂商所提供的工具和基础架构 。有些工程师认为这些工具很成功，但是越来越多的工程师感到困惑。因为测试工具不够自动化，测试不够充分，测试不是由数据驱动的，而且测试脚本的生成和数据的合成方法亦有待改进。软件测试自动化的一个解决方案，是开发测试工具，而不是购买基于当前不充分的基础架构的商用工具。

本书所针对的，是与软件工程打交道的人，而且他们想将公司的测试过程实现自动化。使用本书所提供的方法，软件工程师可以更好地了解当前测试工具在测试自动化方面的局限性，以及如何改进当前的测试基础架构，完成全自动化的软件测试。

本书亦可针对想要使用更有效的办法实现软件测试的软件工程师。本书所介绍的自动化测试工具既可以作为独立的软件测试工具，也可以作为商用工具的辅助。

为更好地使用本书，应具备一定的软件开发经验，并且是公司软件测试项目的一员。本书中的解释和例子均易懂而且易于实施。中级到高级的程序员以及那些希望拓展软件开发和测试方面知识的程序员均可使用本书。

本书的内容包括用C#编程的技巧。然后慢慢过渡到开发完全自动化的GUI测试工具。尽管本书中的例代码使用C#，且在微软的Windows平台上写成，但这些工具是从Visual Basic 6工程进化而来的，我就曾经使用这些方法为某测试项目开发了一个Java的测试工具。

本书也可以针对软件开发公司或使用软件的公司的经理人员。正如经济学家所报告的，软件故障每年都造成美国经济的实际损失。大概有一半左右的损失发生在软件生产行业。如果读者是一位软件公司的高级管理员，一定对如何改进软件测试方法很感兴趣。另一半的损失来自于软件最终用户的口袋。如果你的企业或研究所有软件的最终用户，很可能需要一个维护小组来支持合同厂家所提供的软件。对测试方法的了解可有助于你的公司更有效地使用软件。

本书的结构

为了更好地表达清楚本书提出的概念，前两章是软件测试的介绍，当前测试工具以及软件测试者的期望。第3章至第6章集中在开发GUI测试库所需的Win32 API编程和.NET的基础原理。第7章到第14章集中在设计和实现自动GUI测试工具直至最终实现完全的自动化。以下是每一章的简短描述。

第1章，“GUI测试：概述”，描述了软件GUI测试的管理和.NET中可以动态测试软件的技术。

第2章，“现有的GUI工具与将开发的GUI工具的比较”，这一章先简单描述了市场上的一些自动化GUI测试工具和本书所示的测试方法的主要特点。通过研究证明，目前市场上的工具是不充分的。本章的目的是展示为什么需要一种新的改进的测试方法，为测试工具增加更多的测试能力以及如何为新的、复杂的软件项目创建完全自动化的测试。

第3章，“C# WIN32 API编程和测试猴子”，本章描述如何整合Win32 API函数，使它们可以像C#代码一样开发测试工具。本章也介绍了如何开发一个C# API文本浏览器来替代原来的Visual Basic 6程序员使用的老版本的API文本浏览器。新开发的C# API文本浏览器可用于三个方面。第一，它可以生成C#代码，整合那些开发完全自动化的GUI测试工具所需的Win32函数；第二，在本书介绍如何开发测试工具时，它可以供GUI测试工具测试使用；第三，在本书的进行过程中，C# API文本浏览器可以测试工具的自动化程度。本章的最后，使用C# API文本浏览器演示如何生成测试猴子。

第4章，“开发GUI测试库”，指导读者开发GUI测试库，其中的函数可以从接口中寻找GUI对象，并且操作鼠标动作。

第5章，“.NET编程和GUI测试”，介绍.NET技术，包括映像技术、延迟绑定、序列化和XML编程等，主要是在数据存储、数据表达和GUI测试等方面。

第6章，“测试Windows窗体概述”，描述如何手工生成GUI测试脚本。这些手工生成的脚本是将来完全自动化GUI测试的基础。

第7章，“自动化GUI测试工具的体系结构与实现”，本章讲述自动化GUI测试工具的体系结构，帮助读者完成预期的AutomatedGUITest工具的第一个版本。

第8章，“GUI测试验证的方法”，解释如何有效地验证测试结果，以及如何向AutomatedGUITest工程中添加自动验证的代码。

第9章，“测试标号和装饰性的GUI控件”，描述测试标号和其他装饰性的GUI控件的过程。

第10章，“测试带有键盘输入的TextBox控件”，讨论.NET框架中的SendKeys类，以及升级AutomatedGUITest来测试文本框控件。

第11章，“测试RadioButton和CheckBox控件”，告诉读者在软件应用中如何使用RadioButton和CheckBox控件，以及如何测试他们。

第12章，“GUI测试自动化中的菜单单击”。介绍更多的Win32 API函数，用于发现菜单条目和测试菜单条目。

第13章，“用户定义的和基于COM的控件”介绍如何在Microsoft Visual Studio.NET IDE和Microsoft Visual Studio 6 IDE中开发自定义的GUI控件以及如何升级Automated-GUITest以便测试这些控件。

第14章，“测试非.NET应用”。本章提出测试未管理的应用的测试方法。这一章在向测试工程添加了新的代码后，以Notepad.exe为例，对其进行了测试。

关于例子

例程序以C#和Win32 API函数为编程规则。目的是使用编程语言中预先定义好的方法来完成AutomatedGUITest工具工程，避免从头开始。在各章节中有四种类型的例代码。

- 开发C# API文本浏览器的例代码
- 如何使用C#和Win32 API函数的例代码
- AutomatedGUITest工具测试的例工程
- AutomatedGUITest工具工程的例代码

第一类的例代码主要出现在第3章到第6章。第7章到第14章则专注在测试项目的自动化上。这些章节中的例代码属于第三类。第二类的例代码只有三个，用于模拟真实的被测汇编码。他们出现在第3章和第12章，与全书的测试相符合。

除了C# API文本浏览器之外，第3章的代码还开发了一个测试猴子。第5章开发了一个XML文档浏览器，用于显示AutomatedGUITest工具的测试结果。

从第4章以后，每一章节都向GUI测试库和AutomatedGUITest工程添加了内容。在每一章的结尾，示例代码可以编译生成可执行的汇编码。测试工具会达到不同的自动化等级，直至最后，实现完全的自动化。

如何寻找源代码

每一章中的例代码和工程代码都可以从www.sybex.com下载，可以使用本书的书名，作者的名字或者利用ISBN（4351）搜索。这样可以节省录入的时间。同时还包括工程的一个完全编译版，可以允许在软件项目中，立刻使用AutomatedGUI工具。

运行AutomatedGUITest.exe文件时，需要从Chapter14\AutomatedGUITest\bin\Debug目录将已经下载的例代码拷贝到计算机系统。计算机系统的最小配置为：

- Windows 95/98/2000/NT/XP
- 预装的.NET Framework
- 20MB的可用硬盘空间

如果使用的是Microsoft Visual Studio.NET IDE的早期版本（比Microsoft Visual Studio.NET 2003 IDE更早），可能不能直接打开例工程。原因在于，带有.csproj后缀的C#工程文件与早期的Microsoft Visual Studio.NET IDE版本不兼容。如欲使用这些代码，可以按照每章所示的过程创建新的工程，然后把下载的带有.cs后缀的文件加入到自己的工程里。

尽管本书中的示例代码在Microsoft Visual Studio.NET 2003 IDE环境中开发完成，仍

然有其他源码开放的.NET IDE可以免费下载使用：

Eclipsing .NET IBM向开放源码社团发布了Eclipse.Net。该产品可以运行在Windows 95/98/NT/2000/XP平台。用户可以从www.eclipse.org下载Eclipse.Net的部件。下载eclipse-SDK-2.1.1-win32.zip文件后，与Microsoft.NET SDK一起安装。Microsoft.NET SDK可以从msdn.microsoft.com/netframework/technologyinfo/howtoget/default.aspx网址免费下载。然后从Eclipse.NET IDE获取开放源码的C#插件。www.sys-con.com/webservices/articleprint.cfm?id=360网址有一篇文章详细介绍了下载和安装的过程。

#develop 这是SharpDevelop的缩写，是另外一个开放源码的、可运行于微软的.NET平台的C#和VB.NET的IDE。可以从www.icsharpcode.net/OpenSource/SD/Default.aspx下载。

DotGNU Portable.NET 这一开放源码的工具包含一个C#编译器、汇编编译器和一个运行时的工程师。最初的平台是GNU/Linux，也可以运行在Windows、Solaris、NetBSD、FreeBSC和MacOSX平台。用户可以从www.southern-storm.com.au/portable_net.html下载该产品。

目　录

第1章 GUI测试：概述

毫无疑问，在现代生活中，软件工业已经渗透到教育行业以及其他的行业和企业。几乎所有的美国企业，以及世界上绝大部分国家的产品开发、生产、市场、支持和服务等都依赖于软件行业。减少软件开发成本，提高软件质量，对于软件行业而言，至关重要。每个公司在发布其产品之前，都在寻找更好的办法来实现软件测试。

软件测试领域的改革已经改进了编写测试脚本和生成测试用例的技术，并且引进了针对单元测试、白盒测试、黑盒测试和其他类型测试的测试自动化技术。在多年的进化过程中，软件测试希望可以合并上述测试技术，成为自动化测试。IBM的Mercury Interface和Rational Software软件以及Segue是目前主要的工具厂商。这些工具旨在加速软件开发的生命周期，在产品发布之前找到尽可能多的错误，减少软件开发成本，从而提高应用程序的可靠性。软件测试工具的目的，是生成测试脚本来模拟用户运行正在开发的软件。通常，测试工程师会接受工具厂家的培训，学习如何手工编写测试脚本，或使用“捕获/回放”过程来记录测试脚本。不论是编写脚本还是记录脚本，都是繁重的工作，而且易出错。当前的工具在如何减少软件测试中手工重复劳动方面，有很大缺陷。

近来，企业在购买了商用测试工具后，测试工程师没有办法使用他们，因为这些测试工具在测试的基础架构上不充分。这些工具通常不能处理高级软件项目的复杂性，也不能跟踪技术的进步，在识别不同的第三方部件方面，也不够灵活，而第三方部件在当前的软件开发企业中很常见。毫无疑问，利润良好的公司一定会有一些商用测试工具不知道的商业秘密。这些不充分性使得美国每年因为软件中的缺陷损失595亿美元，因为当前的测试手段（Tassey 2003）不能发现这些缺陷。测试工程师和专家正在为自己的公司开发更有效的测试工具。这样，他们就可以按照以下的原则提高当前的测试基础架构。

- 加强集成和互通性测试
- 提高测试用例生成的效率
- 利用完全的自动化技术改进测试代码生成
- 确定某产品是否达到可以发布的标准的严格监测办法
- 可用的性能测量标准和测试的测量过程

据估计，改进后的测试基础架构能够减少三分之一的当前软件错误，从而为美国每年减少222亿美元的损失。为了达到这样一种软件测试基础架构，很多研究人员都发表了他们的研究成果。例如，在我的上一本书“Effective Software Test Automation: Developing an Automated Software Testing Tool”（Sybex 2004）中，就讨论了如何开发可以适应不同的软件编程语言和平台的测试工具的方法。那本书可以实现完全的自动化测试工具。它使整个软件测试过程实现自动化，从数据生成、测试脚本生成到显示错误。在以下几个方面打破了当前测试工具的瓶颈。

- 主动寻找要测试的部件，而不是利用捕获/回放技术来记录测试场景。这种办法可以使用测试脚本组织一个应用程序的完整测试。据说，如果在测试脚本开发之后使用测试用例，可以更有效地发现错误。因此，测试人员无须再花时间编写测试脚本，而是去研究和创建多个测试用例
- 有效生成可以测试某DLL或EXE中所有成员（构造器，属性和方法）的一个脚本，包括从基础类中继承的成员。该测试脚本中的真实的和完整的对象可以在集成测试和回归测试中重复使用。因此，可以减少一个软件项目中测试脚本的数目，使得测试脚本的管理变得容易
- 使用自底向上的方法，在测试高级模块时，有效地重用低级模块中已建好的测试脚本。这样可以使用户在进行集成测试时，避免磨损（stubbing）和使用假的对象
- 使用开发者使用的语言自动编写测试脚本。测试人员不必学习一种新的脚本语言，开发者也可以明白测试脚本。这样可以加强测试者和开发者的合作性
- 使用UML（统一建模语言）集成自动化脚本生成。有了详细设计之后，测试脚本可以在项目开发过程的早期生成。测试脚本可用于回归测试，并且在开发的生命周期中，在集成测试中重用。因为测试可以在无人看管的情况下，每夜进行，所以可以保证新开发的代码在每个阶段至少可以被测试一次
- 无须编辑和调试，自动编译运行生成的测试脚本。如果在开发工程当中，软件的体系结构设计发生变化，可以立即自动生成新的精确的测试脚本

然而，因为软件产品的每个部件中图形用户接口（GUI）的独特性，开发高效的自动化GUI测试工具所需的技术和编程方法需要单独的一本书。我们会使用在前一本书中对非GUI测试实现自动化时使用的方法，来开发完全自动化的GUI测试工具；也就是说，用户给GUI测试工具一个应用程序，就可以得到错误报告。

GUI测试的独特性

早期的软件应用是命令行驱动的。用户需要记住并且输入某个命令，然后系统完成某些功能。更有效的应用会在屏幕上显示若干可选的命令，让用户输入下一个命令。现在，软件行业已经进入到窗口的时代。所用的应用程序都是靠图形化用户接口GUI来操作的。GUI部件包括窗口、图标、菜单和箭头。用户拖动鼠标和击键来完成各种键盘按键组合。应用程序由鼠标事件和键盘按键事件触发，完成所需的动作。GUI使得软件对用户更加友好。

测试中的软件验证和有效性确认是质量保证过程的主要组成部分。对GUI测试尤为重要，因为这是站在最终用户的角度上进行的。应用程序的功能是通过图形化用户接口来调用的，所以GUI测试可以涵盖整个应用。

在GUI时代以前的自动化测试软件，测试者使用的是由若干行命令做成的测试脚本。程序的运行与屏幕状态无关。测试GUI部件与此不同，而且困难得多，它需要脚本重新分配输入流，单击按钮，移动箭头，还要击键。脚本还需要专门的机制用于记录软件的响应和动态状态的变化。将响应与变化与预期基线进行比较之后，脚本可以报告错误。

有一些专用的工具可以测试基于GUI的应用，他们各有千秋。测试者希望他们在进行GUI

测试时，能够考虑到平台的不同，满足识别GUI部件、测试脚本的自动化、同步、结果验证、易管理等需求。当前的工具受平台影响很大。例如，微软的32位Window操作系统（Win32）是目前的主流开发平台。在Win32上开发的测试工具就不能用在其他平台上，如UNIX、Linux和Macintosh等。

目前广泛使用的生成GUI测试脚本的方法是捕获/回放技术。这种技术要求测试者通过鼠标事件和键盘击键进行繁重的与GUI 交互的工作。脚本记录这些事件，然后以自动测试的方式进行回放。这种方法产生的脚本通常是硬编码的。如果在测试时需要不同的输入，测试脚本需要重新生成。如果GUI部件处在开发过程当中的话，使用这些测试脚本进行回归测试也不现实。如果想对所有的GUI部件生成所有可能的测试用例，从而实现完整的测试，这种方法也很困难。人工与GUI交互通常会产生错误，这样一来，捕获/回放过程就容易记录很多冗余和错误的击键和键盘输入。

基于当前的测试技术，GUI测试自动化总是需要人工编写、编辑和调试脚本。一方面，工具厂商不断地告诉测试者他们的工具功能多么强大；而另一方面，测试者必须面对大量的技术挑战，使得测试工具能够与被测试的应用（AUT）协调工作。众所周知，研发工程中，GUI部件会被修改或重定义。生成的测试脚本不能及时跟踪这些设计变化。

与非GUI软件测试相比，基于当前测试工具的GUI测试基础架构的售价通常比较昂贵，而且在测试过程中需要维护。通常只有一部分GUI部件是自动测试的，其他部分仍然需要人工测试。在以后的章节中，我会说明GUI测试的不充分性，并且讨论改进当前测试基础架构的方法。

开发自动化GUI测试工具

使用捕获/回放技术记录的原始的GUI测试脚本，在成功运行的情况下，能够完成明显地鼠标和键盘动作。但是，就算是能够成功地运行，捕获的结果不能验证由GUI事件所引起的企业函数调用（通常是一些非GUI模块）是否正确。还必须要测试复杂图形输出。GUI测试脚本不能测试GUI部件的输入变化。如果没有手工地编辑和调试，这些测试文本无法测试GUI部件是否按要求地被创建和画出。

本书的目的是演示如何使用更加有效的编程方法来开发GUI测试工具。该工具可以帮助用户避免当前测试方法的缺点和缺陷。生成的测试脚本可以捕获正在被测的GUI部件的事件动作和变化，以及由GUI事件引起的与非GUI部件有关的调用。它亦可以验证GUI部件的位置和外观，并且测试GUI部件的输入。事件动作和调用的结果可以从GUI转换成可读的文本格式。这样开发者就可以使用测试结果报告修正发现的错误。

为改进测试基础架构，本书讨论了开发GUI测试工具的方法，这样的测试工具可以生成无暇的测试脚本。这些脚本所包含的函数可以测试GUI部件的不同方面。GUI部件有所改变后，这些脚本也不会作废。如果增加或减少了GUI控件，只要将改变的应用程序输入到工具中即可生成新的测试。

在本书中，也讨论了如何开发需要最少培训的测试工具。当前的测试工具的用户接口很复杂，用户必须经过培训后才能使用。几乎每个GUI测试工具都有专门的编写测试脚本所

用的脚本语言。因为语言不同，开发者无法了解软件测试自动过程。本书中，读者将学习如何开发一种工具，使得编写测试脚本的语言与开发时的语言相同。这样，测试工具、被测应用、自动生成的测试脚本都使用一种语言开发。测试项目更加可读和易于维护，从而使测试者和开发者能够沟通。

本书还讨论生成测试用例的方法，从而使测试用例和脚本组合起来更加有效地发现错误。当前市场上依赖于捕获/回放和逆转工程技术的工具都不能定位GUI部件并自动为之编写相关的测试脚本。因此我们需要能够自动而且彻底搜索GUI部件的方法，并且按照部件中不同的测试事件编写测试脚本和测试数据的方法。

最后，使用这些方法开发测试工具能够节省时间和金钱，发布精确的产品，并且使公司对测试感兴趣。

自动测试的预期值

自动化测试可以明显减少整个开发生命周期中软件测试的时间和资金开销。而且，自动化可以保证测试有规律而且一致地进行，在早期实现错误检测，从而提高质量，缩短产品上市时间。

当前的测试技术在生成可运行的测试脚本之前，不能实现自动化的GUI测试过程。测试者需要使用捕获/回放技术来记录脚本。记录过程需要测试者和GUI之间繁重的交互。如果发生错误，记录过程不能完成，或者在测试中发现了一个错误，测试者必须重复和继续脚本记录过程，直到错误被修正了为止。错误是一个一个地在人工脚本记录过程中被发现的。当记录好的脚本可以运行时，通过重放来发现错误的可能性很有限。

测试者希望测试工具能够主动寻找GUI部件，生成相应的测试数据，使用生成的数据驱动测试脚本的生成和运行。尚没有一个工具能够实现主动地在应用程序中搜索存在的GUI部件。尽管厂家声称他们的工具可以实现数据驱动的脚本生成。测试者仍然需要启动Wizard输入测试数据。Wizard驱动的代码生成过程很费人力，而且容易出错。这样的数据第一次运行时，找错的效率很低。一个测试脚本必须测试很多测试数据表，才能发现错误。测试人员希望使用完全自动的方法生成测试数据的多份拷贝。

商用的测试工具对平台的依赖性很强，通常他们会有自己的测试环境。一般来讲，被记录下来的测试脚本只能运行在被记录的环境中。工具也必须和依赖于工具的脚本语言一起使用。这样的结果使软件测试一般只限在测试者的系统中进行。如果开发者或其他个人需要参与测试，必须从工具厂家购买更多的许可证。这也限制了测试脚本的移植性。

研究人员和工具开发者正在致力于实现完全自动化的测试工具，可以识别GUI部件，并且可以按顺序地生成测试数据和测试脚本。同时也可以用于回归测试。援引“The Economic Impacts of Inadequate Infrastructure for Software Testing”（Tassey 2003）的说法，当前的测试方法亟待改进和修正，有些甚至应该摒弃。在开发测试工具时，我们应该参考专家的意见。能够更有效地定位错误的新工具应该具备以下特性：

- 测试脚本语言应该与开发被测应用的语言相同。用户无需手工单击鼠标，按键盘和记录测试脚本。这样，测试不仅可以在测试者的系统中运行，也可以在开发者的系统中运行。能够在用户的系统中运行测试，有助于应用程序的阿尔法和贝塔版本发布

- 软件测试的管理会变得容易。因为工具可以主动寻找GUI部件，编写测试脚本，因此可以实现测试的最大化，并且减少测试脚本生成的冗余。一个测试脚本能够在最大数目的测试用例中进行测试
- 开发完成的工具拥有开放的体系结构，从而灵活而且易于修改，便于扩展测试功能。可以使用企业内有限的资源升级测试工具。这样，工具总能够跟上技术的进步，满足软件项目的复杂性要求。工具也可以识别第三方的GUI部件、定制的控件以及测试数据和脚本生成中的一般对象
- 回归测试可以完全自动化。测试运行是测试过程中最无聊、机械、重复和死板的一部分。有了这个工具，回归测试过程可以在无需人工干涉的情况下昼夜运行
- 测试结果将使用人们广泛接受的格式（如流行的电子表格程序、XML文档或HTML格式）报告和存储。当发现错误时，报告可以在源代码中指出问题。用户无需培训即可理解和使用该报告改正错误。甚至可以启动互联网错误跟踪系统

自动测试队伍

一个正确地自动化后的测试过程可以在短时间内运行更多的测试用例，发现更多的错误。Charles Schwab & Co，美国一个主要的股票经纪公司报告说：一个需要52小时手工测试的典型的系统测试，经自动化后，仅需3个小时。自动数据处理公司（ADP）报告说，自动化使得他们的测试时间减少了60%。有研究表明，被工具用户发现的缺陷从10%提高到50%。许多公司使用自动化测试工具使得产品的质量更高。

而另一方面，在一项针对250个公司进行的调查中，只有35%的测试者在工具安装一年后，仍然使用自动化工具，原因在于测试的不充分性。为避免回到繁杂而且耗时的手工测试方法，很多公司培训员工开发更有效的自动化测试。图1.1所示为测试队伍的公司级图表。一个效率高的测试队伍必须包括对产品质量和测试自动化程度感兴趣的高级管理者。开发人员应主动分享和传达测试事件的知识。工具开发人员应基于手工测试者和工具用户的经验来开发更有效的测试工具。阿尔法测试者独立工作，但应该与其他测试者共享测试脚本（手工的或自动的）。

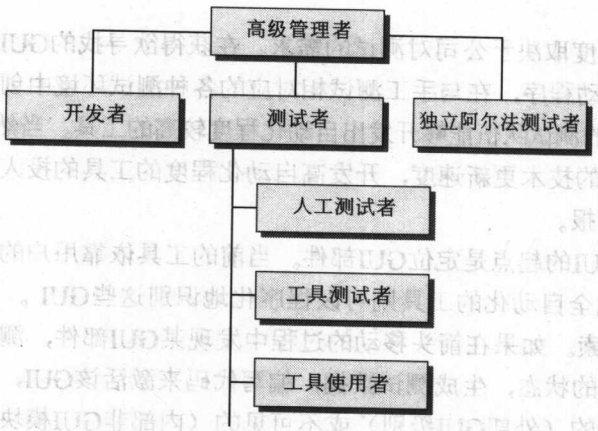

图1.1 测试队伍的公司级图表，包括高级管理者，支持的开发人员，测试人员和阿尔法测试人员