论文题目：基于驱动的软件自动化测试研究与应用

专　　业：计算机软件与理论

硕 士 生：庞晓剑

指导教师：毛明志 副教授

摘　要

软件自动化测试是软件测试体系中的一个重要分支，是实现测试自动化战略的重要组成部分。实施正确合理的自动化测试能分担手工测试的工作量，特别是对回归测试，功能测试等，从而保证软件质量，缩短开发周期，降低成本。

本文融合了数据驱动和关键字驱动的核心思想，提出了一个基于驱动的自动化测试框架。该框架具有数据与脚本分离、减少总测试脚本数、增加测试脚本的健壮性和易维护性，界面元素名与测试内部对象名的分离，测试描述与具体实现细节的分离等优点，对自动化测试具有一定的实际意义。并且根据此框架实现了一个自动化测试工具，能有效进行 GUI 和非 GUI 的自动化测试，改进了单元测试的效率，自动生成测试数据和 XML 文件存储，自动生成测试脚本和测试结果报告，简化了测试用例的生成和维护过程，其基于驱动的测试脚本生成方法有效解决了基于捕捉/回放的测试脚本的缺点。该工具可以作为功能测试和回归测试方面的辅助工具，这对于减少手工测试的工作量和测试时间，缩短软件开发周期和提高软件质量具有实际的意义和价值。

关键词：自动化测试，数据驱动，关键字驱动，GUI

Title: The Research and Application of Driven-based Software Automated-testing

Major: Computer Software and Theory

Name: Pang Xiao jian

Supervisor: Ass. Professor Mingzhi Mao

ABSTRACT

Software automated testing, an important part of automated test strategy, is one of the important branches of software testing architecture. It can lighten the workload of manual testing, especially the regress-testing and function testing, therefore it can guarantee the software quality, shorten developing cycle and reduce cost.

This paper integrates the core of data-driven and keyword-driven thinking, and proposes a framework on driven-based and automated testing. The framework has some advantages, such as separating data from script, reducing the amount of testing script, enhancing the strong and easy-maintenance of test script, separating interface element name from testing inner object name, testing description from implement detail. It is practical for automated testing. Moreover, the paper implements an automated testing tool which can test graphic user interface and non graphic user interface effectively by this framework. It can improve the efficiency of unit testing, and create test data and XML store file automatically, as well as scripts and result reports, and simplify the creation and maintenance of testing use cases. The creation method of driven-based testing scripts solves the problem of play/rollback testing script. This tool, which can be used for the associated tool in function testing and regress testing, is practical and valuable for reducing the workload of manual testing and testing time, guaranteeing the software quality and the reducing the software develop cycle and enhancing the software quality.

Key Words: Automated testing, Data-driven, Keyword-driven, GUI

第1章 绪论

1.1 研究背景和意义

软件测试是为了提高被测软件的质量、对被测软件进行设计和维护，是软件质量保证的关键步骤。

传统的软件测试流程一般是先在软件开发过程中进行少量的单元测试，然后在整个软件开发结束阶段，集中进行大量的测试，包括集成测试和系统测试。随着开发的软件项目日益庞大和复杂，传统的软件测试流程不可避免地给带来项目进度难于控制、项目管理难度加大、对于项目风险的控制能力较弱、软件项目开发费用超出预算等问题[22]。

引入自动化测试是解决上述问题的有效途径。自动化测试可以提高测试效率，使测试人员更加专注于新的测试模块的建立和开发，从而提高测试覆盖率；其次，自动化测试使测试资产的管理数字化，并使测试资产得以在整个测试生命周期内复用，这个特点在功能测试和回归测试中尤其具有意义；此外，通过测试流程的自动化管理可以提高测试过程的有效性。根据 Oppenheimer Funds 的调查，在 2001 年前后的 3 年中，全球范围内由于采用了自动化测试手段所实现的投资回报率(ROI)更高达 1500%。

有效的自动化测试需要良好的自动化测试工具，良好的测试工具需要良好的自动化测试框架支撑，自动化测试框架可以使得测试脚本的维护量减至最少。目前流行的自动化测试工具主要采用了基于数据驱动或基于关键字驱动的自动化测试框架，测试脚本或程序的编写采用测试人员编写、测试工具利用逆工程编写、捕获/回放测试场景等方式，改善了软件测试的效率，但无论哪种方式都需要调试，并不是完全的自动化测试工具。

随着软件规模的不断扩大，软件测试的功能测试和回归测试阶段的工作量不断加大，此时完全的自动化测试工具变得逐渐重要。实施正确合理的自动化测试

能分担手工测试的工作量，特别是对回归测试，功能测试等，从而保证软件质量，缩短开发周期，降低成本。

1.2 主要研究的内容

本文以软件自动化测试为线索。通过对自动化测试和自动化测试框架的研究，以及在流行的自动化测试工具基础上，实现一个完全自动化测试工具，用来作为它们在功能测试和回归测试方面的辅助工具。具体研究内容如下：

(1) 对软件自动化测试技术进行分析和总结。阐述了软件自动化测试现状、特点、适用范围和相关自动化测试工具。

(2) 分析目前主要的自动化测试框架，重点分析比较了数据驱动自动化测试框架和关键字自动化测试框架，提出本文的自动化测试框架的理论基础。

(3) 根据本文提出的基于驱动的自动化测试框架，设计和实现一个基于驱动的自动化测试工具。

1.3 创新点

本文的创新点如下：

1) 在融合数据驱动和关键字驱动思想的自动化测试基础上提出了一个 ADTF 框架，它具有脚本与数据分离、界面元素与测试内部对象名分离、数据驱动测试脚本生成等特点。

2) 开发了一个用于 GUI 和非 GUI 测试的自动化测试工具 ADT，该工具改进了单元测试的效率，自动生成测试数据和 XML 文件存储，自动生成测试脚本和测试结果报告，简化了测试用例的生成和维护过程，其基于驱动的测试脚本生成有效改进了基于捕捉/回放的测试脚本的缺点。

1.4 论文的组织结构

本文共有六章组成：

第一章 绪论，介绍了论文的背景和意义、研究内容、创新点和论文的组织结构。

第二章 阐述了自动化测试技术的研究现状、适用范围、特点、工具。

第三章 介绍了目前主要的自动化测试框架，并详细分析关键字驱动和数据驱动的自动化测试方法，提出本文所需的理论基础，

第四章 提出了基于驱动的自动化测试框架 ADTF，并分析了其特点。

第五章 详细设计和实现基于 ADTF 的软件自动化测试工具 ADT。

第六章 总结和展望。

第 2 章 自动化测试综述

2.1 研究进展

“自动化测试”就是使用软件工具来代替手工进行的一系列测试动作，从而验证是否满足需求。自动化测试的目的是减轻手工测试的工作量，以达到节约资源(包括人力、物力等)，保证软件质量，缩短测试周期的效果。通常是使用脚本或者其他代码驱动应用程序。这一切可以通过可视用户界面完成，也可以通过直接命令(从客户端发向服务器，以模仿浏览器发送的命令)完成。

自动化测试技术按其机制可以分为侵入式和非侵入式，侵入式测试技术采取某种方式修改内部代码或者控制其运行环境;而非侵入式测试技术用于监视和检查软件，而不修改软件内部结构或者代码。

自动化测试主要采用自动化工具提供的测试脚本完成针对目标应用程序的测试。测试脚本是某种采用特定语言编写，并在特定系统环境下实现的代码，根据测试功能的复杂程度，测试脚本可以是需要借助其他语言进行解析的文本，可以是简单的“批处理”命令，也可以是相对复杂的类似于 Tcl 语言的功能强大的脚本语言程序片段。当前使用的测试脚本主要由三种方式产生[23]:

第一种方式是人工编辑测试脚本，与普通的编程原理类似，人工编辑测试脚本就是采用某种特定的编程语言，编写一系列能够在特定环境和平台下运行的代码(如批处理脚本)，然后运行这些代码，达到自动测试的目的。

第二种方式是测试工具利用“面向对象的软件逆工程”技术自动产生测试脚本，这种方式主要用于单元测试自动化技术中。对于采用 OO 方式设计的目标应用程序，单元就是程序中的各个类，针对这些类的单元测试采用这种技术生成测试脚本，首先要提供被测单元的源程序代码，在该代码单元内人工选择需要测试的方法，产生一个或者多个测试脚本。

第二种方式就是“录制一回放”脚本技术，这种方式属于大多数 GUI 自动化测试工具都使用的主流测试自动化技术，主要是由测试人员在测试工具平台环

境下，手工对目标应用程序进行一次用例测试，由测试工具记录下手工操作的步骤和对象，作为测试脚本运用到以后的多次机械重复测试中去，以提高测试效率，减少重复测试工作量。

自动化测试理论方面，目前有许多软件自动化测试过程的模型[24,25,26,27,28,29]，包括软件自动化测试生命周期模型[30]。

Krause 为自动化的软件测试提出了四级成熟度模型[24]，将软件测试成熟度模型和卡内基梅隆大学的 SEI(软件工程研究所)的软件过程成熟度模型联系起来，把成熟度模型分为:附属级自动化，初始级白动化，主体级自动化和优先级自动化，从概念上描述了自动化，但它仅仅描述了在一些测试组织中注意到的一些问题。

Dustin, Rashka 和 Paul 合作公布了自动化测试生命周期方法学(Automated Test Life Cycle Methodology, ATLM)[30] 这是一种经过调整的结构化方法学，能确保自动化测试的成功实现。定义了:自动化测试的决策;自动化测试的引入;测试计划、设计和开发;自动化测试的执行和管理。这个模型从管理和控制的角度描述了自动化测试。

Powers 提出了一些实际建议，从一般意义上讨论了编程风格、命名规则以及用于编写自动化测试脚本的其他惯例[27]，对自动化测试构建和实现很有帮助。powers 还综合论述了数据抽象的原则，对自动化的软件测试来说，数据抽象是数据驱动方法的基础。

Strang 的数据驱动自动化测试方法[28]，使用脚本框架从测试数据知识库中读取所需要的值，这需要用到包含了输入及其预期操作的数据文件。使数据抽象不仅存储了字面值，还存储预期的结果。这种方法既适合手工数据生成，也适合自动化的数据生成。Archer Croup 的控制同步数据驱动测试(Control Synchronized Data Driven Testing CSDDT)是它的一个例子。

基于框架的模块化测试脚本设计技术是数据驱动测试的补充。它实际上是一种结构化的程序设计，用于脚本设计和构建。Zambelich 提出的功能分解(functional decomposition) [29]进一步增强了基于框架的测试概念，将所有的测试用例减小到最基本的任务，并且编写可以相互独立执行任务的脚本。

2.2 自动化测试特点

自动化测试具有以下特点：

(1) 提高测试效率

手工测试是一个劳动密集型的工作，并且容易出错。引入自动测试能够用更有效、可重复的自动测试环境代替繁琐的手工测试活动，而且能在更少的时间内完成更多的测试工作，从而提高了测试人员的工作效率。

(2) 降低对软件新版本进行回归测试的开销

对于现代软件的迭代增量开发，每一个新版本大部分功能和界面都和上一个版本相似或完全相同，这时要对新版本再次进行已有的测试，这部分工作多为重复工作，特别适合使用自动化测试来完成，从而减小回归测试的开销。

(3) 完成手工测试不能或难以完成的测试。

对于一些非功能性方面的测试，如：压力测试、并发测试、大数据量测试、崩溃性测试等，这些测试用手工测试是很难，甚至是不可能完成的。但自动化测试则能方便地执行这些测试，比如并发测试，使用自动化测试工具就可以模拟来自多方的并发操作了。

(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，所以可以进行重复的测试，使得每次执行的测试具有一致性，手工测试则很难做到这点。

(5) 更好地利用资源

将繁琐的测试任务自动化，可以使测试人员解脱出来，将精力更多地投入到测试用例的设计和必要的手工测试当中。并且理想的自动化测试能够按计划完全自动地运行，使得完全可以利用周末和晚上的时间执行自动测试，每日构建(Nightly/Daily Building)技术日渐普遍。

(6) 降低风险

自动化测试能通过较少的开销获得更彻底的测试效果，从而更好地提高了软件产品的质量。

总而言之，自动化测试通过较小的开销可以获得更多的测试，提高软件的质量。

2.3 自动化测试的适用范围

Peer 给出了对各类型测试进行自动化的参考[31]，如表 2-1

表 2-1 自动化测试在不同类型测试下的作用

技术	描述	备注
单元测试/组件测试	该测试工作通常是开发人员的职责，很多不同的方法能够被使用，比如“预测先行”，它是一个测试框架，开发人员在编写代码前编写不同的单元测试。当测试通过时，代码也就完成。	通过使用正式的自动化单元测试，不仅能够帮助开发人员产出更加稳定的代码而且能够使软件的整体质量更加的好。
冒烟测试/构建版本测试	冒烟测试是一般验证被测系统的功能测试用例的集合。冒烟测试背后的思想是确保基础是可以工作的，以便更大的测试工作能够开始。	在构建过程能够确保构建已经为测试准备好时，冒烟测试通常是自动化的运行。
功能/集成测试	这里测试的工作关注在验证不同的组件之间的集成上。	这些类型的测试通常是复杂测试的基础，大量的边缘测试被合并以制造出不同的错误处理测试。
系统测试/用例测试	这种测试是通过执行用户场景模拟真实用户使用系统以证明系统具有被期望的功能的测试。	不需要进行自动化的测试、安装测试、安全性测试通常是由手工完成的，因为系统的环境是恒定不变的。
回归测试	回归测试实际上是重复已经存在的测试。通常如果是手工完成的话，这种测试只在项目的结尾执行少数几次。	这里完全有潜力应用自动化的测试。能够在每次构建完成后执行自动化的回归测试，以验证被测试系统的改变是否影响了系统的其他功能。
性能测试	性能测试包括以下不同测试形式： 负载测试 压力测试 并发测试	如果没有自动化的测试工具，将无法执行通过模拟用户的负载实现的高密集度的性能测试。

可见重复性比较高的回归测试，以及手工测试难以完成的性能测试是自动化测试最有潜力应用和最能体现价值的地方，而早期的单元测试中如果引入自动化，也将大大提高开发出的代码质量。

表 2-2[31]对何时使用自动化测试和何时使用手工测试进行了一个概要的总结：

表 2-2 自动化测试和手工测试的使用

使用自动化测试	使用手工测试
项目没有严格的时间压力，具有良好定义的测试策略和测试计划：知道要测试什么和知道什么时候测试； 对于自动化测试拥有一个能够被识别的测试框架和候选者能够确保多个测试运行的构建策略； 多平台环境需要被测试； 拥有运行测试的硬件； 拥有关注在自动化过程上的资源； 被测试系统是可自动化测试的。	没有适当的测试过程； 没有一个测试什么，什么时候测试的清晰的蓝图； 刚加入项目组，并且还不是完全的理解方案的功能性或者设计项目组成员或者整个项目在时间的压力下； 在团队中没有资源或者具有自动化测试技能的人； 没有硬件。

2.4 自动化测试工具

自动化测试需要不同类型的自动化测试工具进行支持。现在市面上有各种软件测试工具，包括 IBM 的 Rational 系列、Mercury 的 Mercury Interactives 系列，Segue 的 Scgue Software, Compuware 的 DevPartner 系列和其他厂商的工具，还有一些开发源代码的测试工具，如 Ant、JUnit, JProbe 和 Cactus 等，如表 2-3 所示。

表 2-3 当前主流测试工具的优点和不足

工具	厂商	主要特性和缺陷
WinRunner	Merury Interactive	特性：捕获/回放自动化测试工具； 记录时上下文敏感度较好。 缺陷：需人工产生测试脚本； 测试脚本太大会导致不可预测的错误； 可移植回放验证较弱。
Robot	IBM Rational	特性：功能测试和回归测试工具；

		在用户指示可自动产生测试用例和测试脚本； 缺陷：数据池使用麻烦，一定要在 Script 中，只能利用界面定义一个脚本池对象； VP 使用麻烦，测试过程较大时，需录制很多 VP； 录制的脚本，其 FullRec 的名字方式使用不方便。
JTest	parasoft	特性：java 语言的自动化白盒测试工具 可提高代码的可靠性。 缺陷：需手动检查大量的 JUnit 代码 重复代码多； 不支持迭代/敏捷的开发模型； 报告的信息过多，不正确的测试将无法获取 Bug； 静态的技术； 价格昂贵。

根据在测试过程中何时以及如何使用分为如下几类[23]：

(1) 测试管理工具：测试管理工具对测试需求、测试计划、测试用例、测试实施进行管理，并且测试管理工具还包括对缺陷的跟踪管理。测试管理工具能让测试人员、开发人员或其他的 IT 人员通过一个中央知识库，在不同地方就能交互信息。代表的 IBM Rational 公司的 Test Manager、Mercury 公司 Quality Center 的 Test Director 等软件。

(2) 白盒测试工具：白盒测试工具一般是针对代码的内部逻辑流程和结果进行测试，测试中发现的缺陷可以定位到代码级。根据测试工具原理的不同，又可以分为静态测试工具和动态测试工具，具体描述如下：

1) 静态分析工具:静态测试工具直接对代码进行分析，无需编译运行。静态

测试工具用于两个方面：代码评审，一般是对代码进行语法扫描，找出不符合编码规范的地方；静态结构分析，根据代码流程图计算复杂度评价代码设计，生成模块的调用关系图等。静态测试工具的代表有 Telelogic 公司的 Logiscope 软件、PR 公司的 PRQA 软件。

2) 动态分析工具:动态测试工具需要运行代码，一般采用“插桩”的方式，向代码生成的可执行文件中插入一些监测代码，用来统计程序运行时的数据及覆盖率。动态测试工具的代表有 IBM 公司 Rational 的 Purify Plus 、Compuwar 公司的 DevPartner。

(3) 黑盒测试工具：也称为功能测试工具，通过自动录制、检测和回放用户的应用操作，将被测系统的输出记录同预先给定的标准结果比较，功能测试工具能够有效地帮助测试人员对复杂的企业级应用的不同发布版本的功能进行测试，提高测试人员的工作效率和质量。其主要目的是检测应用程序是否能够达到预期的功能并正常运行。主要有 GUI 测试驱动和捕获/回放工具和非 GUI 测试驱动工具。功能测试工具的代表有 IBM 公司 Rational 系列的 Robot 和 Functional Tester 以及 Mercury 公司的 QuickTestProfessional 和 Winrunner。

(4) 性能测试工具：这类测试工具的主要目的是度量应用系统的可扩展性和性能，是一种预测系统行为和性能的自动化测试工具。在实施并发负载过程中，通过实时性能监测来确认和查找问题，并针对所发现问题对系统性能进行优化，确保应用的成功部署。负载压力测试工具能够对整个企业架构进行测试，通过这些测试，企业能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。性能测试工具的代表有 Mercury 公司的 LoadRunner、Segue 的 SilkPerformer 以及开源的 Apache JMeter。

(5) 测试辅助工具：这些工具本身并不执行测试，例如它们可以生成测试数据，为测试提供数据准备。

针对不同类型的测试选择专门的自动化测试工具具有重要意义：例如一个工具用于脚本测试和界面控制，一个工具用于向服务器发送直接请求，一个工具用于验证应用程序接口 API 的功能，一个工具用于验证代码是否标准，以及其他一些认为必要或需要的测试工具。

第3章 自动化测试框架

3.1 自动化测试框架的指导原则

为了最大限度的使用测试框架，需要使其可复用、可管理。下面所描述的是开发自动测试框架所需要遵从的一些基本的指导原则[25]：

(1) 要重视自动化测试，一旦决定要自动化，就必须当成主要的工作来做；

(2) 测试设计过程和自动化测试框架必须作为两个单独的实体来开发；

(3) 测试框架应该独立于应用程序；

(4) 测试框架应该易于扩展、维护和增强；

(5) 测试策略/设计词汇表应该独立于框架；

(6) 测试策略/设计应该对测试者隐藏测试框架的复杂性。

3.2 常见的自动化化测试框架

本节中讨论的几种常见的软件自动化测试框架是理论框架。测试小组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具体的自动化测试框架的基础。

3.2.1 测试脚本模块化框架

测试脚本模块化方法[32]需要创建能够代表待测试应用程序(application-under-test)的模块，零件(Section)和函数等小脚本。然后用一种分级的方式将这些小脚本组成更大的测试，实现一个特定的测试用例。在本章中所提及的所有框架中，这种框架应该是最容易去掌握和使用的。就在一个部件

前面构建一个抽象层以掩藏应用程序其他的部件方面，它是一种很著名的编程策略。它把应用程序从在部件中的修改中隔离开来并规定了在应用程序设计中的模块性。为了提高自动化测试组合(test suite)的可修改性和可测量性，测试脚本模块化技术应用了抽象或者封装的原则。

3.2.2 测试库框架

测试库构架框架[32]与模块化测试脚本模块化方法非常相似，有着同样的优势，但是它把待测试的应用程序分成过程和函数，而不是脚本。这种框架要求创建代表待测试应用程序模块，零件和函数的库文件(SQA Basic libraries, APIs, DLLs 等)。然后这些库文件被测试用例脚本直接调用。

3.2.3 关键字驱动测试框架

关键字驱动和表驱动测试是一种独立于应用程序的自动化框架[7]，它们是可以互换的术语。这种框架要求开发于用来运行的自动化工具，驱动待测试应用程序和数据的测试脚本代码相独立的数据表和关键字。关键字驱动测试看上去与手工测试用例非常相似。在关键字驱动测试里，应用程序的功能特性被写在表格和各个测试的详细指引里了。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。

这个自动化测试框架是由 SAS Institute 的 Carl Nagle 开发的，它是一个关键字驱动的自动化测试框架[25]。

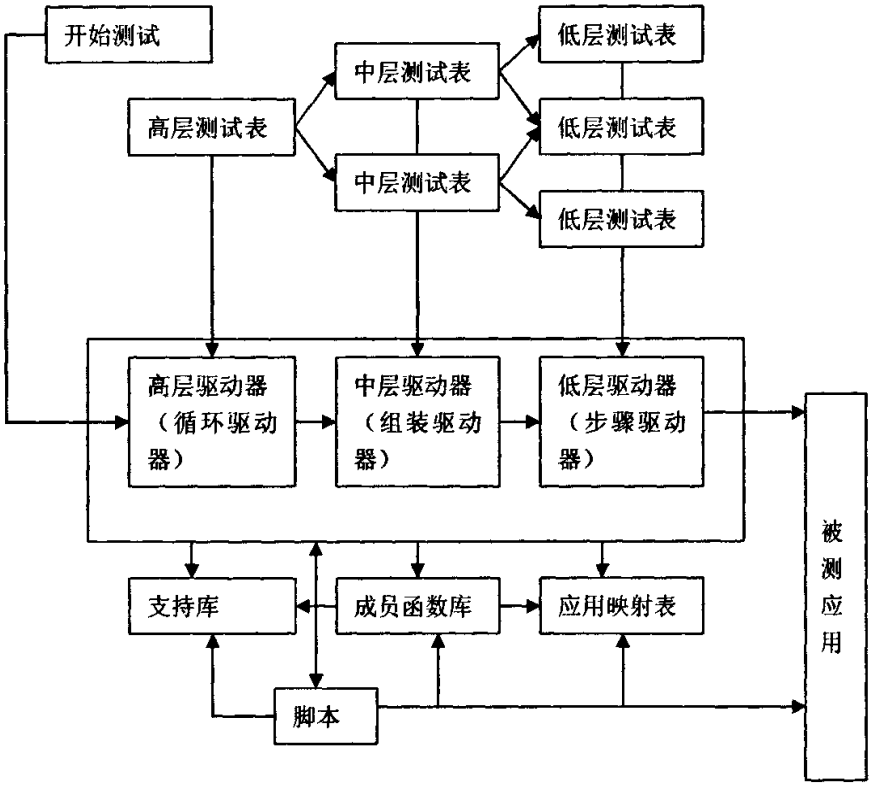


图 3-1 软件自动化测试框架的结构

图 3-1 是这个自动化测试框架的结构。这个框架主要由核心数据驱动引擎、成员函数、支持库和映射表组成。测试首先由初始脚本开始执行，这个脚本把高层测试表传递给高层驱动器，高层驱动器在处理这些表的过程中，遇到中层测试表就调用中层驱动器，中层驱动器处理中层表时也作类似的处理。当低层驱动器处理低层表的时候，它试着使应用与测试保持同步。当低层驱动器遇到对某一个成员的低层关键字指令时，它判断这个成员的类型并调用相应的成员函数模块来处理这个指令操作。所有这些元素都要依靠映射表中的信息，它是自动化模型和被测应用之间的桥梁。该框架主要组件如下：

(1) 应用映射表(Application Map)

映射表是我们的自动化模型中最关键的组件之一。在进行功能测试设计之前，测试人员首先对应用中的每一个对象定义一套命名规范，并利用映射表把这些名字和自动化工具识别的对象名联系起来，使工具能准确地定位和操纵对象，我们

的脚本只需进行单点维护。在上面的例子中,如果按钮的名字或显示文字发生了变化,那么脚本中所有涉及这个按钮的地方都要进行修改。

(2) 成员函数(Component Function)

成员函数是实现用户对界面对象操作指令的函数。

(3) 测试表和核心数据驱动引擎

测试表分低层、中层和高层。低层测试表指定了测试的每一步指令的细节,这些指令都是直接作用在界面对象上的,是无法再细分的指令。中层测试表把低层表组装起来执行更多有用的任务。同一个低层表可以用于多个中层表,所以我们应该开发尽可能少的低层表,然后把它们按照不同的目的组装起来,实现最大的重用性。同样的,高层测试表把中层表组装起来,形成一个测试循环,每个循环是一个完整的测试用例,可以定制不同类型和数量的测试。

(4) 支持库(Support Libraries)

支持库是一般目的的程序和工具,例如文件处理、字符串处理、缓冲处理、数据库访问、日志记录工具等,它们为自动化模型提供最基础的支持。

使用自动化框架进行测试开发的流程如图3-2所示,首先由测试设计人员定义高层循环表,每个高层循环表定义了一个测试的执行周期,其中包括多个待实现的子任务(即高层关键字)。接着设计中层表,每个中层表对应着高层表的一个子任务,在中层表中根据实际测试经验对高层表中的每一个子任务进行详细扩展。

一般高层和中间层的测试设计都十分抽象,所以不需要引用具体的界面元素。但是当我们向最终的低层测试设计靠近时,就必须要引用具体的应用元素了。这时,需要制定应用映射表。先要对界面元素进行命名,要取有意义和易识别的名字,一旦名字被脚本所使用,就不应该再修改;然后用手工捕捉的方式获得每一个界面元素的类型和识别方法,将新的内部名和识别方法对应起来就形成了映射表。映射表的物理格式可以是分界的文件、数据库表、电子数据表或其他可能的形式。

一般的自动化工具已经为我们提供了常用成员类型的成员函数,而在建立映射表的过程中,我们可能会发现新的没有被成员函数处理的成员类型。因此,需要为每个新成员类型实现一个新的成员函数模块,或决定已有的成员函数能否最佳

处理这些新类型，这就进一步完善了成员函数库和低层词库。

最后，设计步骤表。步骤表将上一层的关键字细化为具体的、对界面元素的直接操作行为。在这个过程中，我们会用到在映射表中定义的成员名和成员函数中定义的关键字库对步骤表进行填空。如果测试设计师发现成员函数中缺少必须的关键字、行为或命令时，脚本开发人员就要对相应的成员函数模块、驱动器命令和支持库进行扩展，或找到其他一些解决方法来提供这些功能。

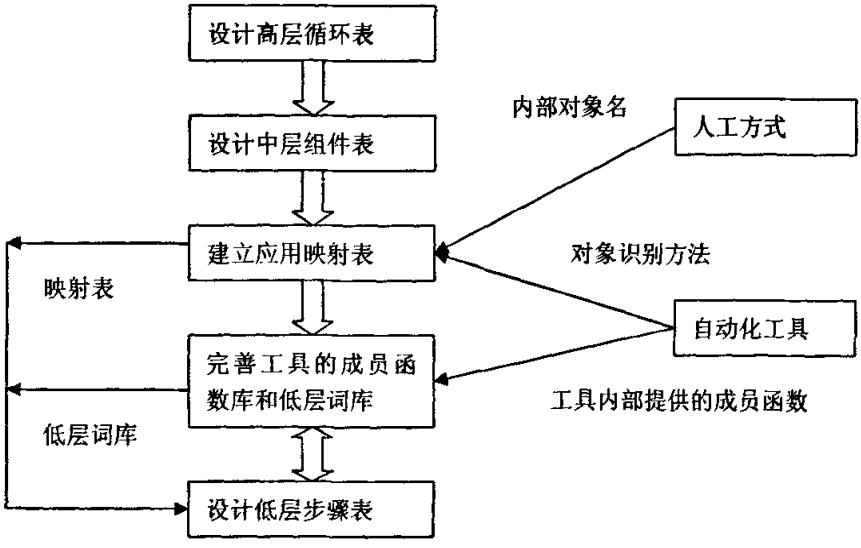


图 3-2 自动化测试开发的流程

关键字驱动测试框架优缺点如表 3-1 所示：

表 3-1 关键字驱动测试框架优缺点

优点	缺点
(1) 把传统测试脚本中变化的与不变的东西进行了分离，这种分离使得分工更明确，并且避免了它们相互之间的影响； (2) 容易维护和使用，不仅维护简单，而且可以反复应用于各种应用，并能长期发挥它的作用。 (3) 测试人员不需要录制测试脚本，只需要	关键字驱动的自动化测试模型的开发和实现与传统的测试流程相比困难且耗时。因为，它需要完全地将测试和自动化工具以及应用程序本身的变化隔离开来，需要增强自动化工具所提供的组件功能，例如，错误纠正、避免和数据同步。

设计测试脚本。	

3.2.4 数据驱动测试框架

数据驱动(Data - driven) 测试框架[28, 32]指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、cvs 文件、excel文件、DAO对象、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。

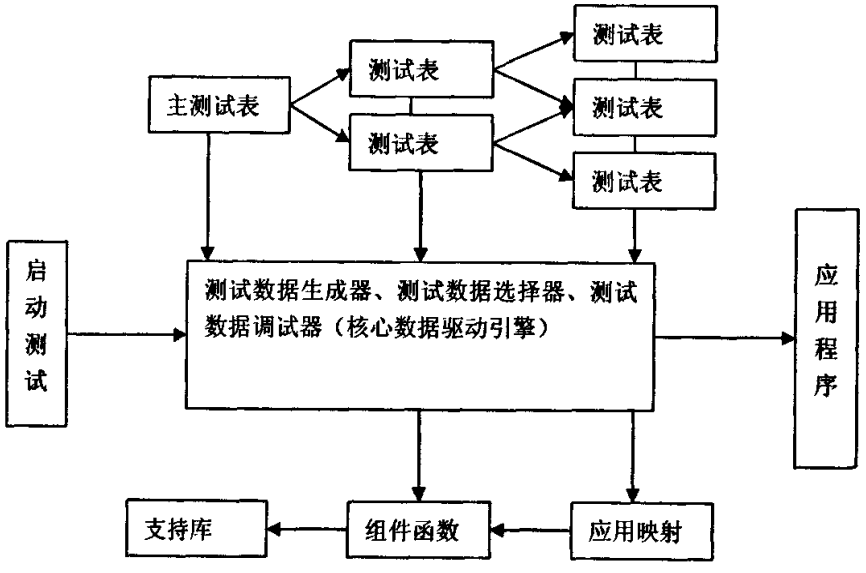


图3-3 数据驱动框架

图 3-3 是数据驱动框架的结构。数据驱动测试框架主要有 4 个部分组成：核心数据驱动引擎、测试表、组件函数、支持库和应用映射等，如图 3.3 所示。核

心数据驱动引擎驱动每个测试表，并对测试数据的产生及选取和控制数据的调度加以控制。测试表中包含了所有用来完成测试所需的相关信息。组件函数中包含了一些既定的功能。当驱动引擎遇到一个指定组件的命令时，会判断组件的类型并激发相关的组件函数来处理任务。支持库中提供了通用的例程。应用映射中包含了 AUT 中的所有对象的结构及其之间的关系。

在被测的应用程序中，每个被测的窗体对应一个测试文件，而每个测试文件可以映射到一个测试表。应用程序的主窗体映射到唯一的一个主测试表。核心数据驱动引擎首先处理主测试表，当主测试表激发了下一级测试表时，核心驱动引擎挂起当前任务，转向下一级测试表处理。核心驱动引擎处理完当前测试表时，返回上一级挂起的任务，继续处理测试表。在处理每一级测试表时，核心驱动引擎会保持应用程序与测试的同步，主要组件有测试表和核心数据驱动引擎，具体描述如下：

(1) 测试表

数据驱动测试是一种数据被包含在输入测试数据文件中，并且数据控制自动化测试脚本执行的流程和动作的测试。测试数据记录以文档的形式包含在测试表中，测试表包含测试数据和控制数据。测试数据进行必要的各种类型的测试，而控制数据引导测试脚本到达合适的位置并指示要执行的动作。

每个被测 AUT 的窗口都对应一个测试表，它是独立文本文件，包括记录类型、窗口代码、对象代码、功能代码、预期目标、测试数据、测试结果等字段。通过填写测试表文件，可以实现基本的业务规则测试、GUI 测试、属性测试和输入数据测试等。具体的描述如下：

1) 记录类型

被核心数据驱动引擎读取的测试表应该首先提供记录的类型，它指明下一步应该怎样处理这些记录。

2) 窗口代码和对象代码

这 2 个字段用来指定在 AUT 中将要选择哪个窗口和对象，这是必须设计的与应用程序相关的代码。在本文中使用窗口和对象的标题来选择窗口和对象。

3) 动作代码:这个代码表示要对目标窗口中的对象执行的动作

共分为 3 类动作：操作动作、同步动作和检测动作

4) 预期目标和测试结果：

这个值被用作错误检测码。

5) 测试数据:这个字段被用作 AUT 的直接输入来代替用户可能键入的数据或做出列表选择。

6) 测试类型、测试次数和测试权值

这 3 个字段被用在核心数据驱动引擎里的控制数据调度器中。

(2) 核心数据驱动引擎

核心数据驱动引擎利用支持库、组件函数和应用映射对主测试表和其他的测试表进行驱动。在驱动每个测试表时，首先从应用映射中导出当前测试表的映射，利用控制数据调度器选择出需要调度的控制数据，进而跳转到相应位置执行。在执行过程中，可以利用测试数据产生器和测试数据选择器来生成相关的测试数据，具体描述如下：

1) 控制数据调度器

调度器首先采用随机算法，随机产生控制数据。结果表明部分控制数据没有被执行，部分被反复执行，出现分布不均的现象。为了避免控制数据分布不均衡，为每个控制数据加上测试次数。这样，控制数据调度器不是无目的地随机选择控制数据，而是选择测试次数最低的控制数据访问，避免了分布不均匀。但实验结果表明控制数据调度序列明显呈周期变化。为解决控制数据调度的次序性，在测试次数最低的所有控制数据中随机
选择，这样有效地防止了控制数据调度的次序性。

2) 测试数据生成器和测试数据选择器

进行测试的目的就是尽可能多的发现错误，而不同的错误被发现的概率是有差别的。逻辑错误最难被发现，要发现他们需要最多的计划和资源。所以，对软件进行深入的测试，不仅需要巧妙的设计、构造，还要执行那些能够检验软件逻辑的测试用例。测试数据选择器的工作原理是按照既定的算法，从测试数据生成器生成的数据集合中选取相应的测试数据。

数据驱动测试框架意图减少你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。它的优缺点如表3-2所示：

表3-2 数据驱动测试框架优缺点

优点	缺点
在应用程序开发的同时就可以同步建立测试脚本，而且当应用功能变动时，只需要修改业务功能部分的脚本； 利用模型化的设计，避免重复的脚本，减少建立或维护脚本的成本； 测试输入数据，验证数据和预期的测试结果与脚本分开，存放在另外的数据文件里，利于测试人员修改和维护； 透过判断功能回传值是“True”或“False”，可作错误处理，增加了测试脚本的健壮性； (1) 支持非程序设计测试员，自动化测试开发人员创建数据驱动的测试过程，测试员创建测试数据； (2) 采用测试过程来收集测试结果，并在输入数据的语境中表示测试结果，这样可以简化手工结果分析。	对自动化测试工具里的脚本语言必须非常精通； 每个脚本都会对应多个数据文件，这些数据文件需要根据脚本的功能类别存放在各自的目录中，增加了使用的复杂性； 测试人员除了需要根据具体测试数据维护相应的测试计划，还要将这些数据写入不同需要的数据文件中； 在编辑数据文件时，必须注意测试脚本所需要的传输格式，否则会在处理脚本时产生错误。

第 4 章 基于驱动的自动化测试框架

4.1 自动化测试框架 ADTF

根据上述的方法，本文设计了如图 4-1 的基于驱动的自动化测试框架 ADTF(Automated Driven Test Framework)，它不仅可以用于对 GUI 和非 GUI 的自动化测试，还能自动执行测试结果的验证，并能通过测试用例的修复技术很好地支持了回归测试过程。脚本驱动测试过程执行完毕之后，提供了自动的验证(Verify)过程，有效简化了测试人员的负担。

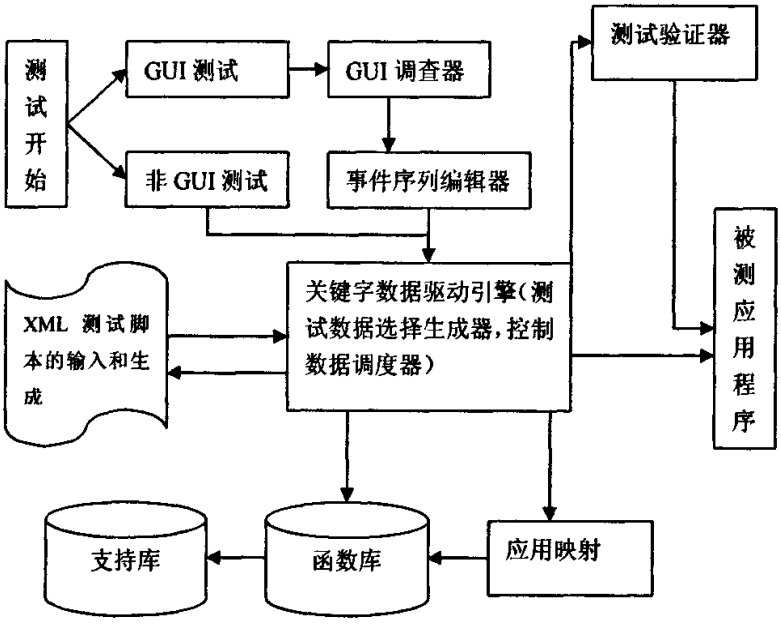


图 4-1 基于驱动的自动化测试框架 ADTF

4.2 ADTF 的主要组件

4.2.1 关键字数据驱动引擎

核心数据驱动引擎利用支持库、组件函数和应用映射对 GUI 和非 GUI 测试的测试表进行驱动。在驱动每个测试表时，首先从应用映射中导出当前测试表的映射，利用控制数据调度器选择出控制数据，然后跳转到相应位置执行。在执行过程中，可以利用测试数据产生器和测试数据选择器来生成相关的测试数据。

4.2.2 控制数据调度器

调度器首先采用随机算法，随机产生控制数据。为每个控制数据加上测试次数以避免控制数据的分布不均衡。控制数据调度器不是无目的地随机选择控制数据，而是选择测试次数最低的控制数据访问，避免了分布不均匀。为解决控制数据调度的次序性，在测试次数最低的所有控制数据中随机选择，这样有效地防止了控制数据调度的次序性。

由于程序在实际的运行过程中，不同的控制数据运行的强度会有所不同。因此，在该算法中引入了测试权值，并在测试权值的基础上，随机为相关测试表选择控制数据，使控制数据产生序列也没有明显的次序性，从而达到预期的目标。

4.2.3 测试数据选择生成器

进行测试的目的就是尽可能多的发现错误，而不同的错误被发现的概率是有差别的。而且逻辑错误最难被发现，因此对软件进行深入的测试，需要进行很好的设计、构造，还要执行那些能够检验软件逻辑的测试用例。测试数据生成器的工作过程如下描述：

(1) 确定并存档测试目标；

(2) 将测试目标翻译成具体的测试需求；

(3) 将测试需求翻译成测试条件；

(4) 构建测试数据；

(5) 导入测试数据。

测试数据选择器的工作原理是按照既定的算法，从测试数据生成器生成的数据集合中选取相应的测试数据。

4.2.4 测试脚本生成器

生成测试脚本。

4.2.5 GUI 调查器

在 GUI 生成测试脚本之前，需要对被测应用 GUI 的结构进行了解，这些信息是框架中 GUI 测试脚本生成器进行工作的基础。

4.2.6 GUI 测试验证器

测试点是在测试流中插入的，它存储着某个特定状态的期待结果，该验证器的作用是在每一个事件执行之后比较测试点与实际的结果状态。

4.2.7 支持库

支持库是一般目的的程序和工具，例如文件处理、字符串处理、缓冲处理、数据库访问、日志记录工具等，它们为自动化模型提供最基础的支持。

4.2.8 函数库

函数库用来存储成员函数或组件函数。成员函数是实现用户操作界面对象指令的函数，一个成员对象的类型对应一个成员函数库。例如对于一个文本框对象，测试人员可能会对它执行多种操作:输入文本、验证文本框的值、验证文本框的某些属性等，实现这些操作行为的函数就被放在文本框的成员函数库中。一般的测试工具都提供了这样的函数，而我们可以在其中加入额外的代码来检测错误、纠正错误和帮助同步，这类代码是实现无人职守的自动化测试所必需的。

成员函数相当于在应用和自动化工具之间提供了一个隔离层，如果没有这个隔离层，自动化工具本身的改变或提高就会影响已有的脚本，但是有了成员函数，我们可以增加 一段修补代码来适应这些变化，转移对测试的破坏。

成员函数关键字和它们的参数构成自动化模型底层的词库,了解了低层词库和映射表,可以建立在它们基础上的测试表

4.2.9 XML 测试脚本的输入和生成

XML 技术是一项将类型和结构置于信息上层的技术。一个 XML 文档就是一组具有一个或者多个命名属性的结构信息项的集合。和 HTML(超文本链接语言)不同的是，XML 语言允许我们自行定义标签，并且对格式有着更严格的要求，因此 XML 文档的格式非常规范。只要定义了一个共同的格式或者结构描述文档，即 XML Schema，应用程序都可以访问符合该 Schema 规范的文档，并将文档的内容以各种灵活的形式(如图形、文本、报表、甚至动画等)表现出来。

由于 XML 存在这些特性，可以设想，如果我们用 XML Schema 定义一个开放的测试脚本数据模型，所有的测试脚本(XML 文档)都遵循这个 Schema，那么第三方就可以开发出各种转换程序，灵活地将测试脚本转换成各种我们所需要的文档，比如某种语言的源代码，测试用例的 HTML 报表等。这种做法，使得

测试脚本和特定工具的耦合度变得非常松散，降低了生成和维护测试脚本的难度，也提高了系统的扩展性。

首先获得待测程序，然后使用一个 XML 测试脚本生成向导来生成测试脚本(或者自行手工撰写)。因为该 XML 文档遵循我们制定的 XML Schema。其他的转换程序就可以很轻松地将 XML 脚本转换成各种文档，比如测试驱动代码、测试报告(HTML 格式或者其他格式)等。对于测试驱动代码，可以使用相应的编译器编译运行；对于 HTML 文档，可以在 Web 上发布；其他的文档，也很容易利用第三方的工具来察看、编辑。如图 4-2 所示。

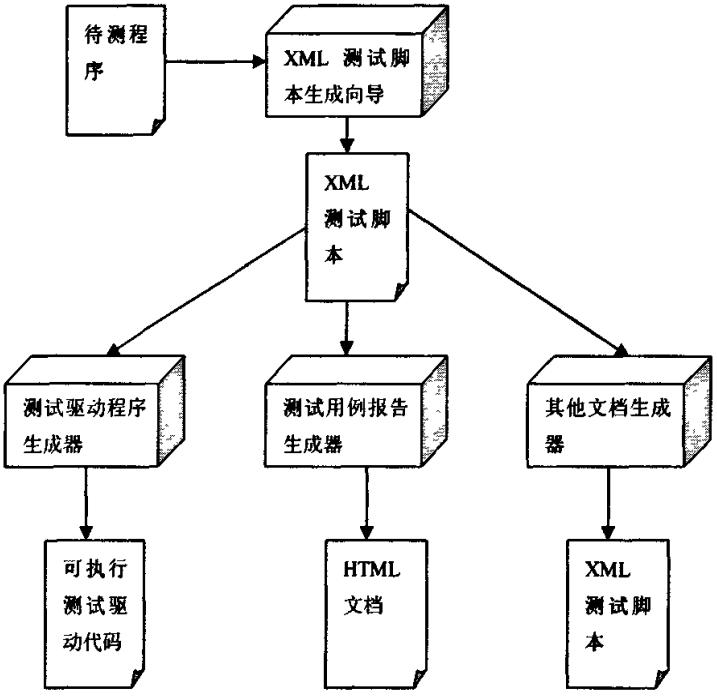


图 4-2 XML 测试模型

这里的 XML 脚本是测试用例的一种更高级的表现形式。它的优点是：利用现有的各种语言的编译器/解释器，简化了自行开发解释器的工作；可以编译执行测试驱动程序，大大提高了执行大规模测试的效率；脚本采用 XML 格式，容易编写和修改；从脚本不但可以生成测试驱动程序，还可以生成其他的各种文档，使用非常灵活；在程序设计语言级别上进行抽象处理。很多语言的概念都是类似的，我们提取出它们的共性，定义一个可扩展的 XML Schema 模型，就可以从

XML 脚本轻松转换到各种程序设计语言的驱动程序，而无需开发各种针对特定语言的脚本模型。

4.3 ADTF 的特点

基于驱动的自动化测试框架 ADTF 具有以下特点：

(1) 融合关键字驱动和数据驱动的核心思想，可以进行 GUI 和非 GUI 测试，以数据驱动测试脚本生成，采用关键字驱动的映射表来识别 GUI 部件；

(2) 使用关键字数据驱动引擎来支持基于驱动的自动化测试；

(3) 界面元素名与测试内部对象名分离。在被测 GUI 程序和生成测试脚本增加一个抽象层，将界面上的所有元素映射成相对应的一个逻辑对象，对这些逻辑对象进行测试。界面元素的改变只会影响映射表，而不会影响测试。

(4) 脚本与数据分离。在测试执行过程可以把所需的测试数据从脚本中提取出来，在运行时测试脚本再从数据存储处读取数据，从而独立维护脚本和数据。

(5) 测试脚本采用 XML 格式，容易编写和修改，提高测试效率。

第 5 章 基于 ADTF 的自动化测试工具

5.1 ADT 工具的测试原理

根据本文设计开发的基于 ADTF 的自动化测试工具 ADT(Automated Driven Test)工具采用了 XML 技术、数据驱动测试的脚本技术、编译原理技术等来实现自动化回归测试，适用于所有用 C#代码写的程序的测试。其测试原理如下：首先选择被测程序，通过对被测程序的信息收集，基于单独的 GUI 和非 GUI 生成测试数据，存入 XML 文档或 Excel 工作表中，并使用测试数据驱动脚本生成，运行测试脚本，产生测试结果报告。

5.2 ADT 工具的模型设计

5.2.1 需求描述

本文采用基于.NET 平台的 Visual Studio 2003 工具和 C#语言开发，并用于测试.NET 平台下 C#开发的软件。C#是一种面向对象、类型安全的开发语言，使程序员快速容易的为 Windows Server System 集成软件产品构建解决方案，它具有代码重用、可靠性、开发效率高、广泛的交互操作性等优点。

自动化测试实际上是弥补手工测试测试的不足，让一部分工作由计算机来自动完成，不需要人工干预的测试。自动化测试工具是通过分析用户的代码自动生成测试脚本来实现自动化测试。所以自动化测试工具要集成代码分析工具、测试脚本产生工具和测试结果比较输出分析工具，通过这些工具的合理运用达到对被测代码的最优检测。ADT 工具中也是通过此种办法来实现测试的自动化。自动

化测试的目的是要在软件开发中尽可能多地发现程序中的错误，这就要求单元测试的测试覆盖率要尽可能高而且要高效。ADT 工具中通过扫描被测试程序，来确定所有的被测方法，从而保证其测试覆盖率。对软件进行自动化测试，测试用例在一定程度上也决定了测试的可靠性和有效性。在 ADT 工具中能够基于每个参数本身的特点，自动产生测试数据值，可以利用开发人员编码时以注释方式用 XML 文档格式写的测试用例。因此，ADT 工具能够自动精确产生测试数据，并节省了测试时间。

因此，在设计和开发 ADT 工具时，具备的功能应该有被测程序信息的提取、测试数据的自动构造、测试脚本的产生及测试结果的比较和报表的生成。整个系统分为 GUI 测试和非 GUI 测试两大模块，非 GUI 测试模块包括：被测程序信息提取模块、测试数据存储模块、测试脚本产生模块、测试结果生成模块。GUI 测试模块包括:GUI 调查器模块、事件序列编辑器模块、GUI 测试验证器模块。

非 GUI 测试功能设计模型图如图 5-1 所示：

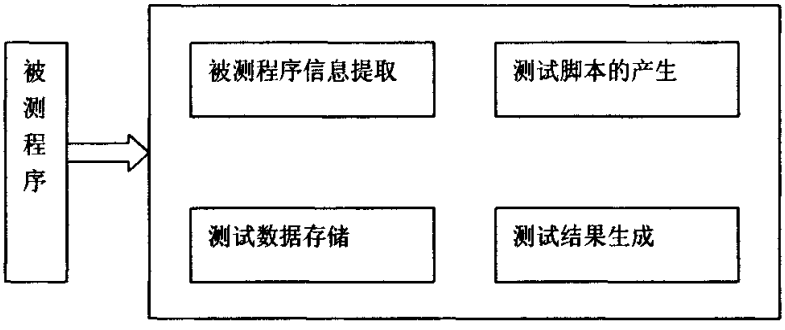


图 5-1 GUI 测试功能设计模型图

GUI 测试功能设计模型图如图 5-2 所示：

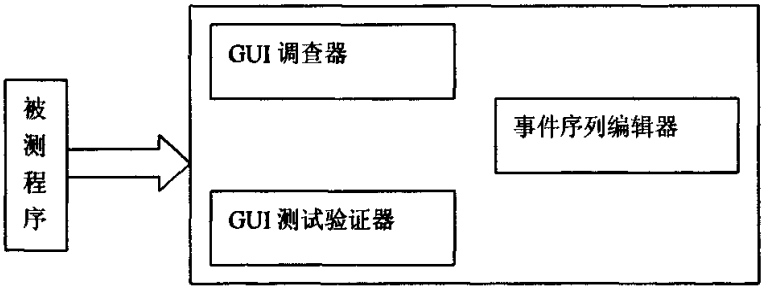


图 5-2 GUI 测试功能设计模型图

非 GUI 测试中被测程序信息提取模块和测试数据存储模块的执行流程图如 5-3 所示：

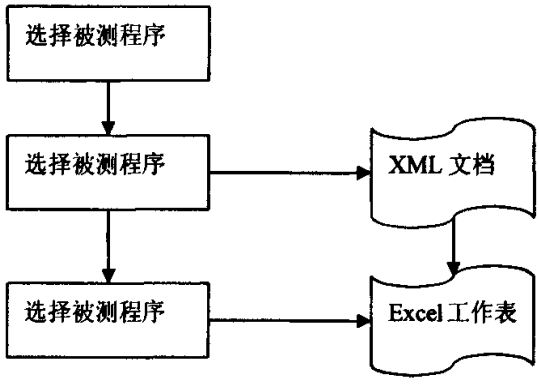


图 5-3 非 GUI 测试执行流程图

GUI 程序的测试过程如表 5-1 所示：

表 5-1 GUI 测试的测试过程

阶段	步骤	测试人员	测试工具
启动被测程序	1	制定被测程序位置	启动被测程序
分析被测程序	2		获取 GUI 就够信息，生成 G-CFG 和 G-Call Tree
测试用例生成	3	指定事件序列	记录测试人员的选择
	4	指定测试点	记录测试点
	5		生成测试脚本
测试用例执行	6		测试脚本驱动测试执行，记录每个测试点的实际值
	7		完成验证过程，生成执行结果
回归测试	8		重复步骤 2，修复测试用例
	9	重复步骤 3，4	重复步骤 3，4
	10		重复步骤 5
	11		重复步骤 6
	12		重复步骤 7

具体描述如下：

(1) ADT 工具为测试人员提供接口，由测试人员指定被测程序，然后由 ADT 自动运行被测程序。本系统接受的是扩展名为.EXE 或者.DLL 的文件。

(2) ADT 工具自动分析被测 GUI 的结构，这是通过对 GUI 的遍历过程得到的。

在遍历的过程中，GUI 所包含的所有对象以及它们的属性都可以获得，并显示出来。测试工具使用这些 GUI 的结构信息创建 G-CFG 和 G-Call Tree。这些结构会在回归测试阶段用于测试用例的修复。

(3) 这是测试用例生成的第一步。此过程获得的 GUI 结构信息会显示给测试人员看，测试人员可以方便地选择每个事件(某个对象的某个事件)，与此同时测试工具会将测试人员所作的选择记录下来。

(4) 这一步是紧接着第一步完成的。因为在指定了一个事件之后，测试工具需要知道对应这个事件的需要验证的测试点。例如，需要在按钮 button1 点击之后验证文本框 textbox1 的文本属性是否为字符串“Hello"。跟步骤 3 一样，测试点信息也被记录了下来。到这里，一个测试用例就已经生成了。

(5) 测试人员提供了生成测试用例的一切信息，下面测试工具按照步骤 3 和 4 中记录的事件序列和测试点的信息生成测试脚本。

(6) 步骤 5 生成的测试脚本会通过驱动器驱动测试过程的执行，与每个测试点相对应的实际结果都会被测试工具自动记录下来。测试执行过程的核心就是测试脚本和底层的动作执行器。

(7) 测试工具自动地将测试点和实际结果进行比较，生成执行报告。该步骤简化了验证过程，同时避免了原来手工验证产生的人为错误。

如表所示，回归测试阶段的步骤基本上重复上面的 7 个步骤，这跟回归测试的本质是一致的。但为了对回归测试过程进行有效的支持，测试工具提供了测试用例的修复功能，使得原来的测试用例经过尽可能小的修改便可以应用于回归测试过程。测试人员只需要依据测试覆盖标准的要求，重复步骤 3, 4, 5，重新生成一部分新的测试用例，从而形成回归测试用例集，用于回归测试。

非 GUI 程序的测试过程如图 5-4 所示：

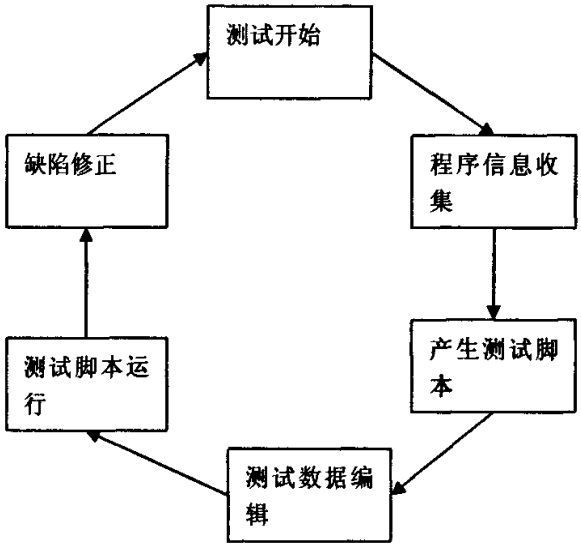


图 5-4 非 GUI 程序的测试过程

5.2.2 主要功能模块

ADT 工具的主要功能模块如下描述：

(1) 被测程序信息提取模块

被测程序信息提取模块的主要功能是：对被测程序利用语法分析器对其进行语法分析，获取程序的结构信息。在测试时，先选择被测程序，选择确定后即开始进行程序扫描，获取信息。实现此功能模块的关键是语法分析程序的构造问题

(2) 测试数据存储模块

此模块的功能主要是完成被测程序的测试数据的存储。即完成将获得的程序扫描信息存入 XML 文档和 Excel 工作表中，其中 XML 文档中存放的是程序的所有信息，Excel 工作表中存放的是选择需要测试的部分的信息。文档和工作表中的记录内容包括类名、方法、属性、参数及其取值等信息。此处利用 Excel 工作表来存放测试数据的好处是：只存放需要测试的内容的信息，这些数据在执行测试脚本前可以任意修改，如对参数值的修改，直接修改相应单元格中的内容即可，如要对一个方法进行多次测试，则可以通过复制工作表中该方法信息所在行的信息即可简单的完成，因此在 ADT 中采用 Excel 工作表是非常方便的，对测试数据的构造通过很简单的操作即可实现。实现该功能模块，主要是要解决在

XML 文档和 Excel 工作表在 Visual Studio.Net 中怎样被操作的问题。

(3) 测试脚本产生模块

测试脚本的产生模块其主要功能是产生测试脚本。AutomatedDrivenTest 工具中采用数据驱动的脚本技术，此技术是将测试输入和期望输出存储在数据文件中(即 Excel 工作表中)，脚本中只存放控制信息。执行测试时，从文件中即 Excel 工作表中读取测试输入。该脚本技术的优点是同一个脚本可以运行不同的测试，可以方便地增加新测试，执行许多具有类似性质的测试，增加新的测试不需要修改脚本。此外，测试者可以将精力放在测试上,而不用过多关心脚本的技术编程问题。

在 ADT 工具中，根据 Excel 工作表中的信息，自动生成变量控制函数、桩函数和驱动函数，并与结果统计模块一同编译，执行，最后输出测试结果。这种方法因不用人工在程序中设定桩函数，所以进一步实现了测试自动化。实现该功能模块主要是解决如何动态产生和编写测试脚本的问题。

(4) 测试结果生成模块

执行测试脚本，其流程图如图 5-5 所示：

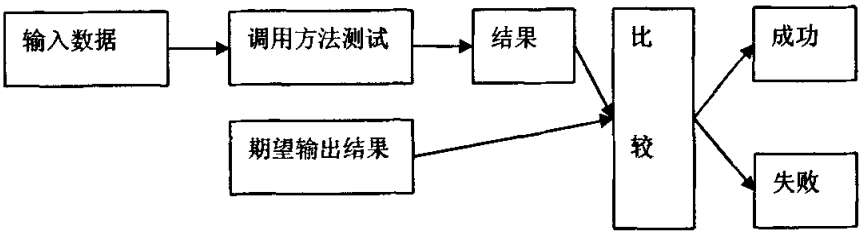


图 5-5 测试结果执行流程图

(5) GUI 调查器

由于 GUI 程序和非 GUI 程序的测试方法有点差异，GUI 程序的测试需要基于捕获/回放，该调查器可以对其 GUI 进行调查，得到其结果信息，这些信息有两个作用：一是用于创建 GUI 控制流图和 GUI 调用树;为了测试用例的修复，需要原 GUI 和修改后的 GUI 控制流图和 GUI 调用树；二是填充事件序列编辑器;测

试人员会在编辑器中指定 GUI 对象。因为，事件是与对象对应的。要想触发一个事件，需要首先指定目标对象。为了识别一个 GUI 对象，ADT 工具使用六个属性，其属性如表 5-2 所示：

表 5-2 识别 GUI 对象所使用的属性及说明

属性名称	说明
句柄(Handle)	一个整形数。由于每个 GUI 对象在实例化以后都会有唯一的一个 Handl。与之对应，当需要查找某个 GUI 对象的时候，就可以通过它进行检索。但是，handle 是一个数字，不便于记忆和理解，所以还需要另外一种更好的策略去方便有效地识别一个 GUI。
标题(Caption)	一个字符串。通常状况下，标题是可以直接发现的，但是有些部件处于非活动状态或者隐藏状态，就无法直接读取。标题是帮助程序精确识别一个 GUI 部件的最重要的属性之一。它较之 Handle 最大的优点在于便于记忆和理解。
标题(Caption)	一个字符串。通常状况下，标题是可以直接发现的，但是有些部件处于非活动状态或者隐藏状态，就无法直接读取。标题是帮助程序精确识别一个 GUI 部件的最重要的属性之一。它较之 Handle 最大的优点在于便于记忆和理解。
类名(Class Name)	一个字符串。每个 GUI 对象都是某个类的子类对象。它决定了这个对象的基本行为和属性特征。因而，类名可以大大增加精确识别 GUI 对象的几率。特别是当对几个具有相同标题的对象进行识别的时候，类名是非常有用的一个属性。
父文本(Parent Text)	一个字符串。前面提到过 GUI 是分等级的，除了主窗体之外，每个窗体都有自己的父窗体，也有可能有自己的子窗体。父文本属性可以将搜索范围缩小在某个较小的范围，从而可以增加精确识别对象的几率。

控件类型(Control Type)	类似于类名，他们的主要区别在于，控件类型是控件这类特殊 GUI 对象的类型，而类名则泛指 GUI 对象的类型。
控件名称(Control Name)	一个字符串。与标题不同的地方在于，标题的作用是对象本身的功能性说明，而控件名称则是对象本身的一个代号。这如同一个人有姓名和对他生平的描述，控件名称是姓名，而标题是他生平的描述。

实现该功能模块主要是解决如何识别被测程序里的 GUI 组件的问题。

(6) 事件序列编辑器

事件序列编辑器也用于 GUI 程序的测试中，一般来说，测试脚本包含测试用例和测试点两个重要元素。该模块主要为测试人员提供接口编辑事件、GUI 部件的信息和测试点信息。

1) 高层事件(High-level Event):系统用户通过键盘和鼠标同GUI系统进行交互，这样就产生了键盘和鼠标事件，这些事件称作基本事件(PrimitiveEvents) 基本事件共有六个属性 :K_P(Key_Press) K_R(Key_Release),M_P(Mouse_Press),M_R(Mouse_Release),M_ M(Mouse_Move) ,M_W(Mouseee_Wheel)，这些事件之所以称为基本事件是因为它们是不可再分的，一个鼠标的点击事件就是由 M_P 和 M_R 事件构成的。基本事件属于底层事件(Low-level Event)，因为它们代表的是键盘和鼠标的操作而不是用户的意图。与之相反的是，高层事件是那些对于具体的 GUI 对象有具体的含义的事件。例如对一个名为 buttonl 的按钮的点击事件(高层事件)就是由 M_M, M_P, M_R 这三个低层事件组成的。测试人员在事件序列编辑器当中指定的事件是高层事件，他们无需关心这些高层事件的低层实现过程，因为测试执行器会按照测试脚本将这些高层事件分解为低层事件去完成的。
2) 测试点(Test Point)，也就是通常所说的检查点(Check Point)，它存储的

是某个特殊状态的期待结果。一个 GUI 状态是指它所包含的所有对象的所有属性的集合，为了测试需要，有时候，测试人员往往只关心某一个或者某几个 GUI 对象的某一个或者某几个属性，那么测试点的内容就比较简单；有时候，测试人员需要对某个事件执行后的系统状态进行全面检查，这就要求测试点包含较多内容。事件序列编辑器必须对两种情况都进行了较好的支持。

实现该功能模块主要是解决如何传递接口编辑事件、GUI 部件的信息和测试点信息给测试验证器的问题。

(7) 测试验证器

对于 GUI 测序来说，验证器的作用在于将测试点同该点的实际值进行比较，然后将验证结果填入执行报告。由于测试人员指定了验证方式，对于不同的方式，测试验证器所要处理的验证对象的数量和频率也是不同的。例如特定对象验证要求测试验证器从第一个事件执行开始每个事件执行完毕就对验证对象的某个属性进行验证；简单的验证要求测试验证器将当前 GUI 的状态与预期的状态进行行验证，当未指定的时候则不进行验证。

对于非 GUI 程序来说，其产生的测试脚本能够引导验证过程并返回结果，这些返回结果验证待测产品是否符合所需的代码逻辑。实现该功能模块主要是解决如何验证测试点的问题。

5.3 ADT 工具的界面设计

图 5-6 是 ADT 工具的界面图：

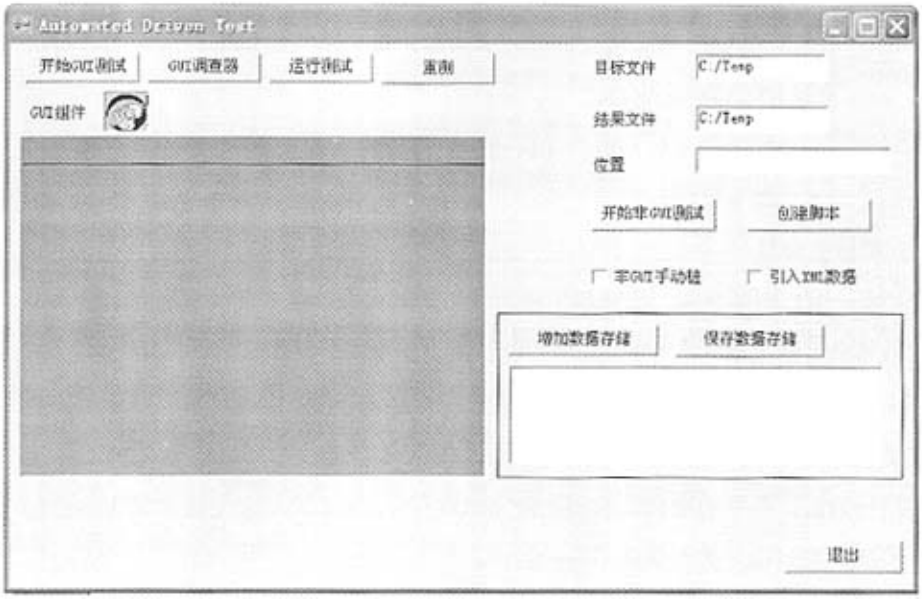


图 5.6 ADT 的界面图

5.4 ADT 工具的类关系图及序列图

ADT 工具的类关系图如图 5-7 所示：

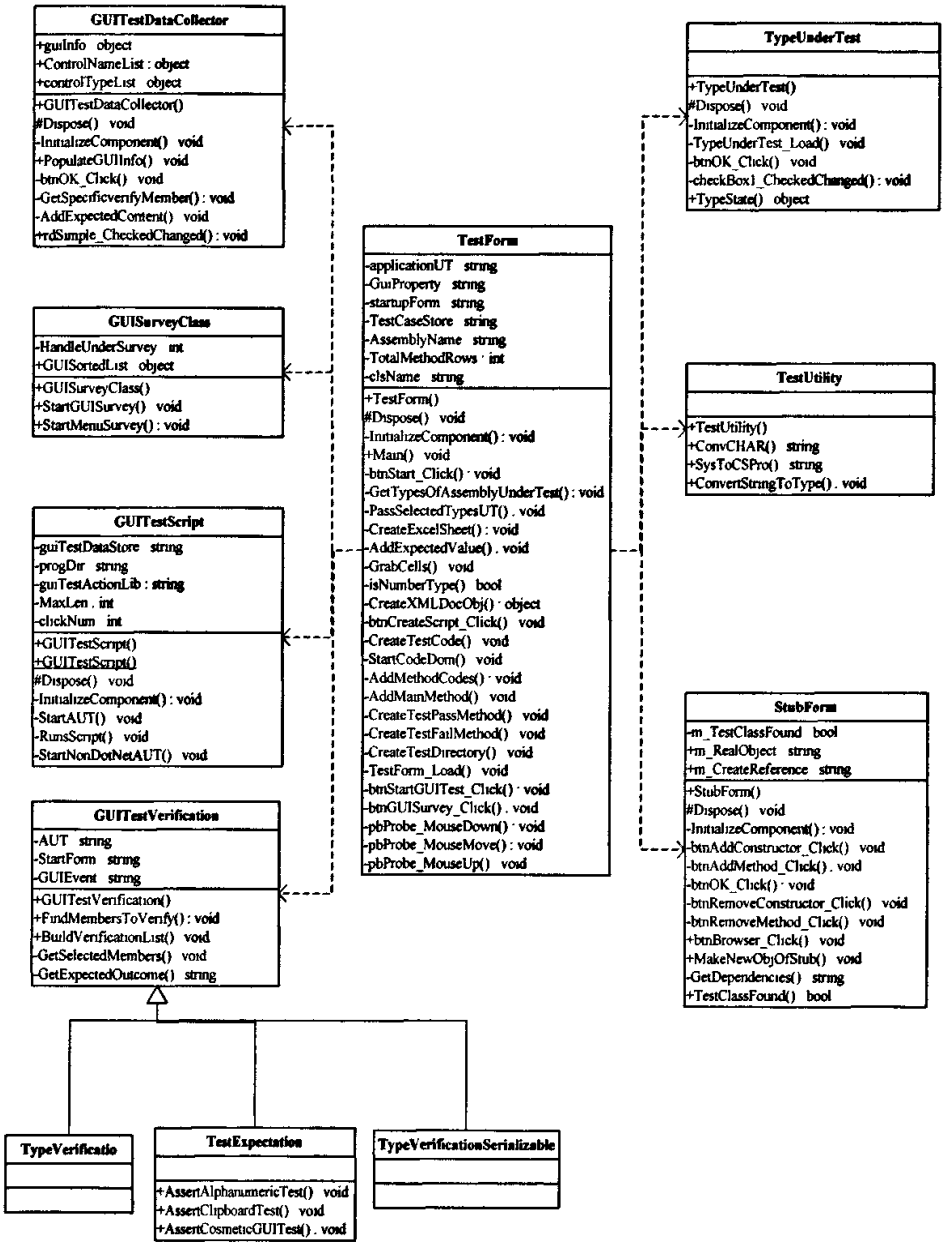
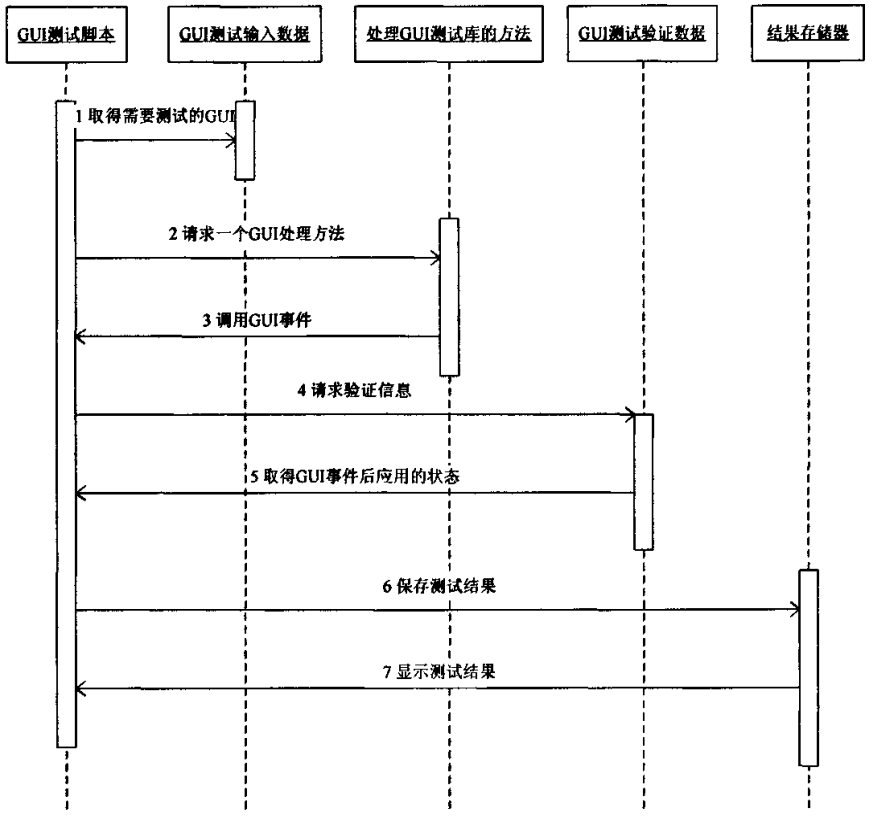


图 5-7 类与类之间的关系

GUI 测试的序列图如图 5-8 所示：



其中2－5步是交互式地进行测试和验证。

图5-8 GUI测试序列图

非GUI测试的序列图如5-9所示：

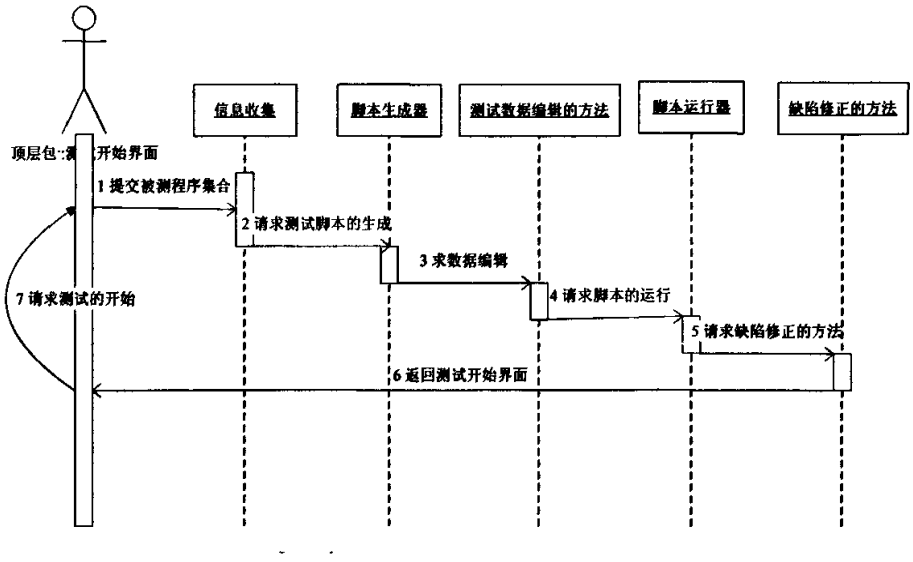


图 5-9 非 GUI 测试序列图

5.5 ADT 工具的实现

5.5.1 ADT 工具技术难点

本文开发的 ADT 工具具有以下技术难点：

(1) 如何对被测程序进行扫描，提取信息；

(2) 如何通过 Excel API 编程和 XML API 编程进行测试数据的存储；

(3) 如何动态产生和编写测试脚本；

(4) 如何动态识别被测程序里的 GUI 组件；

(5) 在事件序列编辑器中如何传递接口编辑事件、GUI 部件的信息和测试点信息给 GUI 测试验证器；

(6) 如何根据事件序列编辑器的信息对测试点进行验证。

5.5.2 被测程序信息的提取

.NET System.Reflection 命名空间的类和方法能够把被测程序映射成它的类型和成员，并提取被测程序集合的信息，包括方法、域、属性、字段等相关细节。抽象类 System.Type 则用于提取被测程序的相关细节信息（如接口、方法、属性等）。ADT 工具使用 Refection 命名空间和类 System.Type 来提取被测程序的相关信息。提取被测程序信息主要包含两个方法：GetAssemblyName() 和 GetTypeOfAssemblyUnderTest()。前者用于指定被测程序集合，后者用于提取被测程序的类型，并置入 TypeUnderTest 窗体。

5.5.3 测试程序存储

在收集被测程序信息时，把获得的被测程序的信息，包括类名、方法名、属性、域、方法的参数等存入 XML 文档和 Excel 工作表。

用 Excel 存储测试数据需要进行基于 C#的 MS Excel API 编程，在进行 ADT 开发时增加对 Microsoft Excel Object Library 的引用以便进行 MS Excel API 编程。

MS Excel 基于自己的层次对象模型组织对象，最高层是 Application 对象，下面包括工作薄、工作表、图标、其他窗口。一个工作薄包括许多工作表，一个工作表包括许多行和列组成的单元格。

该模块的 Excel API 编程主要是类 TestForm 中的四个方法，其名称和作用如表 5-4 所示：

表 5-3 Excel 存储方法名称和说明

方法名称	说明
PassSelectedTypesUT()	收集所提取信息类型
InitMethodInventory()	调用相关方法，处理相关类型
CreateExcelSheet()	Excel 实例初始化和操纵 Excel 表中与数据存储相关的属性

AddExcpectedValue()	测试返回期望值
CrabCells()	根据参数类型和方法修改单元格的内容

CrabCell()方法用于实现 Excel 表格数据存储。该方法声明了 Worksheet 对象、行号、列号、字符串变量 setText、ParameterInf 类型变量 p 五个参数。setText 的值赋给由行号和列号确定的单元格，p 则用于保存被测参数的信息。该方法调用了自动为数据存储赋值的方法 AssignAutoValueToParams()。

InitMethodInventory()方法用于处理被测程序类型的方法。该方法定义了描述被测程序集合的 Assembly 对象 DUTAsm、保存 Excel 数据表的文件名称的字符串对象 XlsFilenameToSave、程序集合中的方法 TotalMethodRows 三个私有参数。

ADT 工具中 Excel 数据存储的布局是：第一列(列 A)列出待测试类型的名称；第二列(列 B)列出所有方法的名称，其余列则列出相应方法的所有参数及返回值。对于需要为测试该方法输入期望返回值的地方，ADT 工具则采用深色作为单元格的背景标出。对于单元格中数据的输入和编辑，ADT 工具则采用注释作为提示，移动鼠标指针到任何参数单元格上，包括期望返回值的单元格，一个注释就弹出来显示参数类型和参数名称。用户可以选择地保留或修改它们。如果用户选择使用工具自动赋值的数据值，那么测试可以在没有数据编辑暂停的情况下完成，以便用户能够编辑测试数据存储，在数据编辑确认之后 ADT 工具继续完成测试脚本的任务，并显示测试结果。

在 Microsoft Visual Studio .NET IDE 中，C#允许开发者添加 XML 文档注释，然后生成 XML 帮助文档。在编写方法的同时编写 XML 注释，建立有效的测试用例，运行测试脚本时 ADT 工具读取存储在 XML 文档中的数据，达到自动化测试的目的。

ADT 中对 XML 文档的解析，是利用 MS XML 解析器解析 XML 文本，XML DOM(文档对象模型)对象提供了 DOM 应用编程接口(API)函数来操作存储在 XML 文档中的信息。该函数是应用程序和 XML 文档之间的桥梁，DOM 主要包括两个抽象概念：一个是树状的层次结构，另一个是用来表示文档内容和结构的节点集合。树状层次包含了所有节点，节点本身也可以包含其他的节点，其优点是可以通过该层次结构来找到并修改特定节点的信息。MS XML 解析器读取一

个 XML 文档，然后把它的内容解析到一个抽象的信息容器中，该信息容器称为节点，这些节点代表文档的结构和内容，并允许应用程序来操作文档中的信息而不需要了解 XML 的语义。XML 文档被解析后，它的节点能够在任何时候被浏览而不需要保持一定的顺序。DOM Document 对象是 XML 编程时最重要的编程对象，它能通过暴露的属性和方法来允许浏览、查询和修改 XML 文档的内容和结构。

该模块的 XML API 编程包括四个方法，其名称和作用如表 5-4 所示：

5-4 XML 相关方法名称及说明

方法名称	说明
MakeUseXMLDataStore()	产生 XmlDocument 对象，确定被测方法的数目，取得需要的参数值
CreateXMLDocObj()	寻找 XML 文档，产生 XmlReader 对象，把信息传递给一个 XmlDocument 对象
ReadXMLTestDataIntoDataStore()	从 XML 文档中提取数据，插入 Excel 工作表中对应的单元格中。
PlaceDataFromXmlDocToExcel()	进行测试时，可以从 Excel 电子数据表中取得数据

5.5.4 测试脚本产生

.NET Code Dom 命名空间具有在运行时建立定制应用程序和输出多种编程语言源代码的能力。该命名空间包含了大量可以描绘任何可能的.NET 类型(如类、接口、结构、枚举或托管等)源代码结构的类型，Code Dom 命名空间能够为这些类型创建各种成员(如属性、方法和事件等)。在成员实现中，Code Dom 命名空间能够编写各种语句，这些类型都是独立于当前编程语言的。Code Dom 命名空间生成的最通用的编程语言是 C#和 VB. NET。Code Dom 命名空间生成的代码在输出之前一直都是处于内存中的。因为 System. Code Dom 命名空间还提供在运行时编译源代码的能力，所以它能够使软件测试实现完全自动化处理。

该模块主要包括生成测试脚本的 CreateTestCode()方法。该方法需要两个参数，前者是给定的待测试程序集 Assembly 对象，后者是用于保存生成的测试脚本的 TextWriter 对象。CreateTestCode()方法生成测试脚本总共可以分为三个步骤，即初始化 Code Dom 对象并为测试脚本编写通用的代码结构、编写针对被测程序生成测试脚本的代码和编写回收测试脚本中所用资源及执行测试脚本的代码。

第一步初始化 Code Dom 对象并为测试脚本编写通用的代码结构。在这一步骤中，它首先调用 StartCodeDom()方法创建必要的 System. Code Dom 对象并为生成的测试脚本指定合适的命名空间名称和类名称。这些 Code Dom 对象将由测试数据存储驱动来编写测试脚本。其次，调用 AddNamespaceRefs()方法为生成测试脚本添加引用程序集。当.NET 语言代码被编译成.NET 通用语言运行时(CLR)运行的代码时，.NET 目标编译器就生成描述包含在可执行文件(EXE 或 DLL)中所有类型的元数据，这个元数据包含了 COM 类型库及版本依赖的特殊信息。当执行或测试应用程序的时候，CLR 在载入前查找程序集以确保每一个依赖它的代码都是有效且合适的。为了实现完全自动化软件测试的过程， ADT 工具需要通过适当的 using 语句添加这些依赖集的引用到测试脚本中。使用 Reflection 命名空间，ADT 工具能够自动地发现依赖集，再通过 Code Dom 对象就可以把这些依赖集添加到测试脚本中。有些程序集引用是每个测试脚本都需要的，如 System 命名空间，它提供所有.NET 的基本对象；System.IO 命名空间，允许测试脚本打开和保存文件；Excel 命名空间，从测试数据存储中读取数据并保存测试结果；System. Reflection 命名空间，为动态绑定提供类和方法。

CreateTestCode()方法中通过调用 MakeFirstClassMethod()方法在测试脚本的命名空间中添加类和方法。测试脚本中的该方法为 StartTest ()，它的返回类型是一个对象，且该返回对象可以被 ADT 工具用于测试高层次模块。再次，它调用 CreateExcelSheetCodes()方法为测试脚本生成一个用于保存数据存储和记录测试结果的启动 MS Excel 电子表格的代码。该方法综合了 Code Dom 和 Excel API 编程。接着，打开 Excel 电子表格之后，它调用 AddExcelHeaderCodes()方法为 Excel 电子表格添加需要的首部信息。第一列(列 A)为待测试方法的名称，第二列(列 B)为测试结果，第三列(列 C)为测试没通过时的错误信息，第四列(列 D)

为该方法的实际返回结果，第五列(列 E)为该方法的期望返回结果，第六列(列 F)及其后为该方法的参数和值。最后，调用 AddMoreMethodCodes()方法继续为任何待测试程序集的测试脚本添加一些通用的代码语句。这些语句主要是给测试脚本的已声明全局变量赋值。

第二步编写针对被测程序生成唯一测试脚本的代码。首先，它从 MS Excel 数据存储单元格中读取待测试类型的名称，然后执行 InitiateTypesUnderTest()方法来初始化该类型。在 InitiateTypesUnderTest()方法中，它首先导入该待测试类型所在的程序集，然后检查该测试类型是类还是接口。根据这个检查，它将添加不同的变量声明到测试脚本中。如果是类，那么就获取该类的名称。如果是接口，那么就先声明一个接口变量，然后获取涉及该接口的类或结构，这些类和结构实现了该接口的抽象成员。

然后，它调用一系列方法来发现类型、构造函数、方法和参数，并为它们在测试脚本中编写代码。首先调用 DiscoverNewTypes()方法确认是否有新的目标测试类型出现。然后，代码检查待测试目标是构造函数、方法还是属性。如果待测试方法或构造函数重载了多次，那么它将使用适当的参数测试所有可能的重载。如果待测试目标是构造函数，那么就通过参数传递及引用来利用先前方法的执行信息为测试脚本添加测试构造函数的代码。一些通过引用传递的参数值将在这个方法的执行期间被修改。从数据存储中找出所有的参数类型，并把它们添加到 AmbiguousOverLoad 类型数组变量中，使用 Type.GetConstructor()方法该数组中的信息重新部署当前构造函数。如果在数据存储中没有发现这个构造函数的参数，那么就测试默认的构造函数。在恰当的构造函数被定位之后，AddTestConstructorCodes()方法使用 ConstructorInfo 对象来反射出参数并调用 AddCstrParameterCode()辅助方法来正确地编码参数，由 Code Dom 编程完成测试特定构造函数脚本代码的编写。如果待测试目标是方法，那么就调用 AddTestMethodCodeI()方法收集测试目标方法(Method)中准确地参数信息并编写测试脚本代码。编写测试目标方法脚本的步骤与构造函数的类似。

第三步编写回收测试脚本中所用资源及执行测试脚本的代码。在编写生成测试不同方法及构造函数脚本的代码之后，需要保存测试结果，关闭打开的文件，和结束测试会话。ADT 工具的实现中采用 AddOtherCodesParth()方法来实现这一

目的。为了保存测试结果，测试脚本在测试运行时会自动地获取日期和时间，然后结合日期、时间和待测试程序集的名称来为保存测试结果指定一个文件名称，从而有利于测试脚本的重新执行及回归测试。

5.5.5 测试结果生成

ADT 工具中的验证是由计算机软硬件和测试脚本执行之间的接口来完成的，它测定以下各项：

(1)待测试类成员是否按预期执行了；

(2)执行是否与指定的硬件和软件配置相兼容；

(3)执行是否捕获异常；

(4)执行是否生成错误提示信息。

最后，验证回答了代码是否按预先定义的系列正确地执行。确认发生在验证完成之后，它包括真实的测试。确认能被定义为在测试结束时评估软件产品以确保正确的输出和顺从软件需求的过程。最后，测试报告呈现验证和确认的测试结果。

ADT 工具的实现中，通过对测试脚本的执行，把两类测试结果插入到 Excel 报告中，其中一类测试结果描述测试执行是否成功，另一类则描述真实执行返回值和期望返回值是否相匹配。该模块主要包括 TestPass()方法和 TestFail()方法。Excel 报告包括了以下的测试结果:

第一列代表被测方法的名称，它继承自测试数据存储；

第二列报告测试是否通过，这是方法调用的结果；

第三列报告错误消息，若测试失败，提示失败的原因。

第四列报告测试后方法的真实返回值，用于检查是否存在指定的期望返回值；

第五列保存 Excel 数据存储中测试人员指定的期望返回值；

第六列开始，测试报告列出了方法执行后所有参数的当前值，用于检查真实返回值的出现条件。

5.5.6 GUI 调查器

当测试人员点击“GUI 调查器”按钮后，它首先将本身的窗体最小化，然后对已经启动的被测应用进行调查。调查过程鼠标指针是按照鼠标步(Mickey Step)由上往下、由左往右移动的，并且速度非常快。调查过程结束以后，AutomatedDrivenTest 的主窗口再次显示在桌面上。调查得到的信息显示在事件序列编辑器中。

该模块主要包括类 GUISurveyClass 和类 TestForm 中的 btnGUISurvey_Click() 方法、setGUIListToTest()方法、dgAvailableGUIs_CurrentCellChanged()方法。

类 GUISurveyCalss 是用来代替捕获/回放过程，以自发的方式在被测 Windows 窗体中进行主动的 GUI 搜索。搜索过程从 GUISurveyClass 的对象初始化开始。该类定义了两个用于搜索的参数。一个是用于获取被搜索的窗体的句柄的整型数变量 HandleUnderSurvey，另一个是保持 GUI 部件的整体性的 SortedList 排序对象 GUISortedList。StartGUISurvey()、StartGUISurvey()、StartMenuSurvey() 是该类的主要方法。它们的说明如表 5-5 所示：

表 5-5 类 GUISurveyClass 方法名称和说明

方法名称	说明
StartGUISurvey()	使构造器接受被测应用的 Windows 窗体句柄，初始化 HandleUnderSurvey
StartGUISurvey()	初始化 GUISortedList 对象，搜索桌面上某个应用程序地每一个像素，驱动鼠标箭头上下左右移动，系统化地、主动地检测所有可用 GUI 部件。
StartMenuSurvey()	进行 GUI 部件的菜单搜索

btnGUISurvey_Click()方法使用 try-catch 字句，传递被测的 Form 对象句柄，以初始化 GUISurveyClass 实例 guiSurveyCls。Try 字句中的语句是用来.NET

Windows Form 应用，catch 语句是测试非 Windows Form 对象应用地。若 guiSurveyCls 成功地初始化，调用 guiSurveyCls 实例中的方法 StartGUISurvey() 将应用中的 GUI 部件收集到 GUISortedList 的 GUITestUtility.GUIInfo 对象。然后调用 SetGUIListToTest()方法将 GUI 标识符设置到控件 DataGrid 中，并显示出来。

5.5.7 事件序列编辑器

在测试编辑器中指定一个事件序列，首先需要指定 GUI 对象，测试人员可以简单地通过双击目标 GUI 对象所在的行选定它。这时，ADT 会要求测试人员指定该事件所对应的测试点。

该模块包括类 TestForm 中的事件 dgAvailableGUIs_CurrentCellChanged()和 dgAvailableGUIs_DoubleClick()和类 GUITestDataCollector。

事件 dgAvailableGUIs_CurrentCellChanged()和 dgAvailableGUIs_DoubleClick() 的作用是给测试人员提供接口来指导工具测试某个指定的 GUI 部件。

类 GUITestDataCollector 的目的是为 GUI 测试验证器进行数据收集。它的界面如图 5-10 所示：

图 5-10 事件序列编辑器界面

类 GUITestDataCollector 主要包括四个方法，其名称和作用说明如表 5-6 所示：

表 5-6 类 GUITestDataCollector 的主要方法名称和说明

名称	说明
rdSpecfic_CheckedChanged()	特定对象验证方法选择，即在测试数据收集的开始，指定某个特定的对象，用于整个测试过程中的每一个步骤之后。选中后，将在 try-catch 字句中，调用方法 GetSpecficVerifyMember()和 AddExcpectedContent（）
rdSimple_CheckedChanged()	简单对象验证方法选择，即将当前 GUI 的状态与预期的状态进行比较，显示测试结果。选中后，调用方法 MakeAllChecked()清除 CheckedListBox 控件中所有已选中的条目和方法 AddExpectedContent()将 GUI 控件添加到多文本框中。
SetSimpleVerification()	启动窗体的名字、验证成员的名字和对象取为参数，调用 CheckedListBox 类中的方法 FindString()使用启动窗体和成员的名字定位索引。
AddExpectedContent()	在多文本框中输入预期结果，与 GUI 组件的内容进行比较验证。

5.5.8 GUI 测试验证器

测试脚本的执行和测试结果的验证是一前一后紧接着进行的。所以，几乎测试脚本执行完毕的同时，测试结果的验证过程也就结束了。

GUI 测试验证器的实现主要由类 GUITestVerification 来实现。该类引用了一些系统命名空间，如表 5-7 所示：

表 5-7 类 GUITestVerification 引用的命名空间和说明

命名空间名称	说明
System.Windows.Forms	在测试 GUI 部件时，可以使类以 Window 以窗体对象的形式传递参数
System.Refection	提供类和方法
System.Collections	收集测试验证所需的信息
System.Xml.Serialization	实现一些序列化的辅助类

类 GUITestVerification 使用了三个序列化的辅助类 TestExpection、TypeVerification、TypeVerficationSerializable，其说明如表 5-8 所示

:

表 5-8 类 GUITestVerification 中的辅助类说明

辅助类名称	说明
TestExpection	确定被验证的成员是否一个 GUI 部件、域或属性； 定义文本验证所需的域、定义验证剪贴板、文件是否存在、整体对象和一些属性所需的域； 方法 AssertAlphanumericTest()用于获取一个布尔型参数 expectedEqual，检查实际结果是否与预期的结果匹配。 方法 AssertClipboardTest()用于剪贴板验证。
TypeVerification	定义被测应用的路径名、应用启动窗体的名字、当前正在验证的已触发的 GUI 事件、ArrayList 等相关变量； 保存验证数据和最终的测试结果前的序列化。
TypeVerficationSerializable	将类 TypeVerification 初始化的变量添加到 ArrayList 域，使得验证器保持一致。

类 GUITestVerification 主要由四个方法实现验证功能，具体步骤如下：

首先，FindMembersToVerify()方法在加载被测应用后使用.NET Refection 命名空间中的 Assembly 类创建新的应用对象，准备一个变量访问应用中的所有的成员，并且修改成员中的 public、private 或 protected 修饰符。然后，该方法使用

foreach 循环提取域和属性，使它们的名字对 GUITestDataCollector 对象可见。

然后，BuildVerificationList()方法将 GUITestDataCollector 对象、整型变量 guiSeq 和 TypeVerificationSerializable 对象接收为参数，将被测应用 AUT 的路径和启动窗体赋值给 TypeVerificationSerializable 对象。然后，初始化新的 TypeVerification，使用 GUITestDataCollector 对象的信息设置新对象的状态。调用 GetSelectedMembers()方法进行新对象与 GUITestDataCollector 对象之间的信息交换。最后，新对象进入 TypesToVerify.TypeList 队列等待验证。

GetSelectedMembers()方法用于从 GUITestDataCollector 对象中收集信息用于验证每个指定的成员。该方法在遍历 TypeVerification 对象的过程中，检查每个成员。如果在成员旁边有选中标记，该方法即将此成员视为需要验证的成员。

最后，GetExpectedOutCome()方法用于从 GUITestDataCollector 类中已经更新的控件和事件获取关于预期结果的信息。

5.6 ADT 工具的应用

ADT 工具分为 GUI 测试和非 GUI 测试，本文将对一个 XML 文档浏览器进行 GUI 测试和对广东省沿海渔民转产转业管理信息系统项目进行非 GUI 测试，该浏览器和项目说明如表 5-8 所示：

表 5-9 被测程序相关介绍

名称	说明
XML 文档浏览器	可以浏览 XML 格式的文件，能够显示 XML 文档的相关信息，比如元素、特征和串值。
广东省沿海渔民转产转业管理信息系统	广东省沿海渔业的管理系统，基于.NET 平台下的 C#开发，采用 Visual Studio.NET 2003+IIS+SQL Server 2000 开发环境。

5.6.1 GUI 测试

对已经编译的 XML 文档浏览器中的 XmlTreeViewer.exe 导入 ADT 测试工具中，进行 GUI 部件搜索，并选择某一个 GUI 部件进行相关验证信息的编辑，然后点击确定自动验证，将当前结果与期望结果进行比较，返回“True”或者“False”如图 5-10 所示：

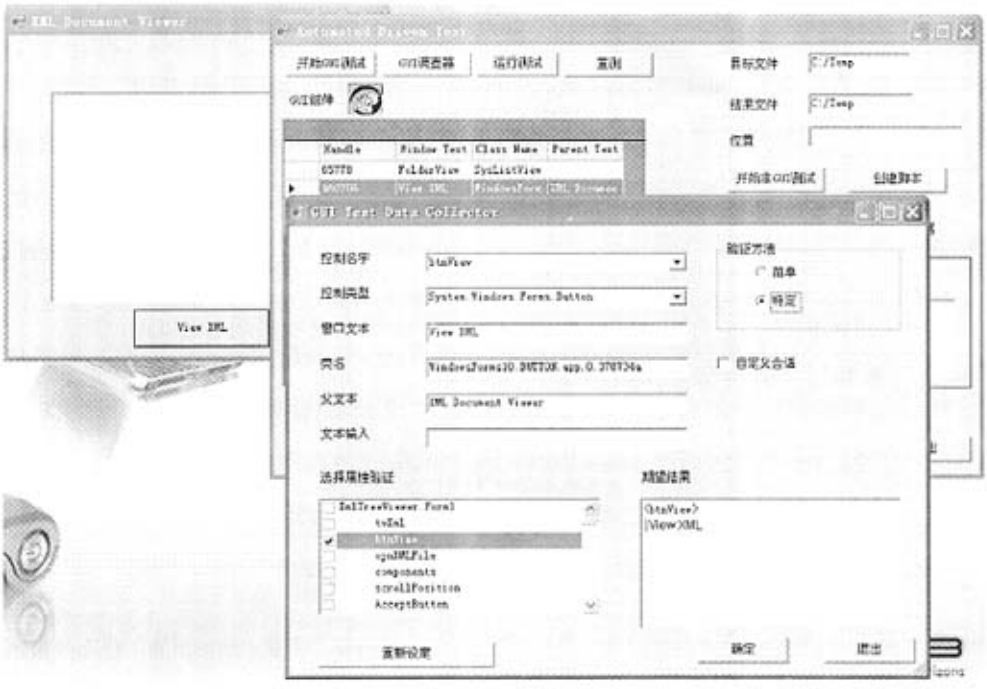


图 5-11 XML 文档浏览器 GUI 测试

5.6.2 非 GUI 测试

点击非 GUI 测试将已经编译的 Fishery.dll 导入 ADT 工具中，选择 Trainee 类进行测试，测试信息及结果如图 5-11、图 5-12 所示：

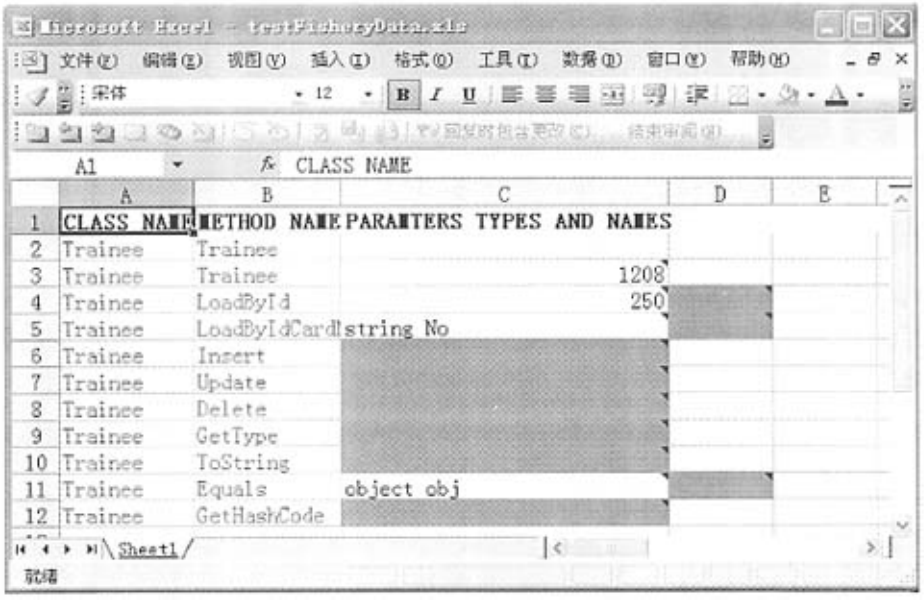


图 5-12 类 Trainee 测试信息

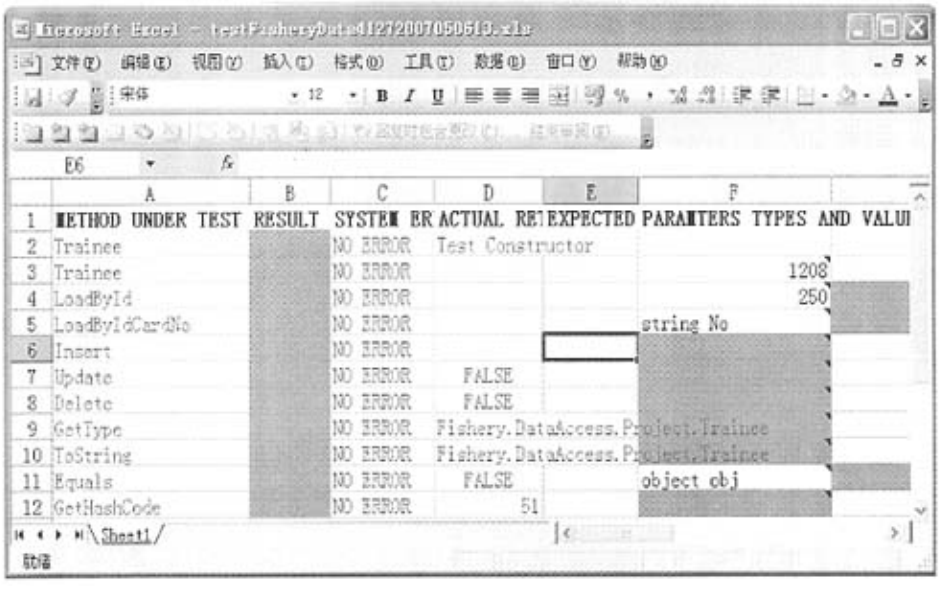


图 5-13 类 Trainee 测试结果

第6章 总结与展望

6.1 总结

本文以软件自动化测试为线索。通过对自动化测试和自动化测试框架的研究，以及在流行的自动化测试工具基础上，实现一个完全自动化测试工具，用来作为它们在功能测试和回归测试方面的辅助工具。

本文所开发的软件自动化测试工具有效改进单元测试的效率，自动生成基于每种参数类型的测试数据并可以自动生成 XML 文件存储，为自动生成桩对象提供了友好的接口，有效改进集成测试，在非 GUI 测试中其基于驱动的测试脚本生成有效改进 GUI 测试中基于捕捉/回放的测试脚本的缺点。

6.2 展望

软件自动化测试是软件测试的一个重要组成部分，它可以完成许多手工测试无法或者难以实现的测试。正确、合理的实施自动化测试，能够快速、彻底的对软件进行测试，从而提高软件质量，节省经费，缩短产品发布周期。

软件测试的自动化是软件测试的一个发展趋势，现在，测试一个项目所需的大量工作应有自动测试工具的支持。手工测试是一个劳动密集型的工作，并且容易出错，它不支持那些可能由自动测试工具完成的相同种类的质量检查。引入自动测试工具能够用更有效、可重复的自动测试环境代替传统的手工测试活动。这项工作本身也提高了测试工程师的工作效率。

自动化测试可以减少测试开销，增加有限时间内的测试。使用自动化测试技术可以更加快速地开发出高质量的软件产品。测试过程中的活动有：标识测试条件、设计测试用例、建立测试、执行测试用例、比较结果。标识和设计活动主要

是智力活动，它们决定测试用例的质量。执行和比较是机械活动比较适合进行自动化。测试验证是检验软件是否产生正确输出的过程，它通过比较实际输出和预期结果来实现。

由于自动化测试的可重用性和测试的一致性好，测试执行效率和资源利用率较高，从而使测试过程变得更加系统化、可控化，亦能被更迅速、更正确地完成。尤其对于需要重复测试的大规模软件，自动测试是更高效的方法，在这种形势下，传统的人工测试已经很难满足要求，发展自动测试成为必然的出路。

在对自动化测试的研究过程中，通过将数据驱动和关键字驱动的方法结合起来分别用于非 GUI 测试和 GUI 测试中，实现了一个用与功能测试和回归测试中地完全自动化测试的工具，有效改进了测试脚本的生成效率和实现了测试数据的自动化生成。但其测试数据的生成算法仍有待改进，在关键字驱动的 GUI 测试中，使用的是无脚本模块化方法，利用 Windows 内置的 GUI 映射表，下一步工作需要改进成脚本模块化和无脚本模块化相结合，从而提高关键字的识别效率。还有继续完善该 ADT 工具。

参考文献

【1】 毛澄映，卢炎生. 构件软件测试技术研究进展 计算机研究与发展 43(8) 2006-10-14
【2】 姚砺，束永安，叶澄清. 面向对象软件覆盖度量的研究和软件测试工具的实现 计算机研究与发展 Vol.39 N0.8 Aug 2002
【3】 梁昆，李明树，梁金能，陈振冲.一种模拟驱动的 Web 应用程序性能测试方法 计算机研究与发展 July 2003
【4】 张广艳，郑名扬，鞠九滨. WebMark：一个 Web 服务器性能测试工具 软件学报 Vol.14 No.7 2003
【5】 杨芙清等. 面向对象软件回归测试技术研究 软件学报 Vol12 No.3 Journal 2001
【6】 姜瑛等. 一种 Web 服务的测试数据自动生成方法 计算机学报 Apr.2005
【7】 冯玉才，唐艳，周淳. 关键字驱动自动化测试的原理和实现 计算机应用 Aug 2004
【8】 黄陇，于洪敏，陈致明，于秀山. 基于 UML 的软件测试自动化研究 计算机应用 July 2004
【9】 赖利锋，刘强. Web 应用程序的一种功能自动化测试模型与实现 计算机工程 2006 年 9 月
【10】朱经纬. XML 技术在软件测试自动化中的应用 计算机工程 2005 年 1 月
【11】朱菊，王志坚，杨雪. 基于数据驱动的软件自动化测试框架 计算机技术与发展 May 2006
【12】何群，陈英. 面向对象语言编译器的自动测试平台 计算机工程 2005 年 7 月
【13】李刚毅，金蓓弘. 自动化回归测试的技术和实现 2006 年
【14】严少清，陈革，万年红. 软件测试自动化管理系统的设计与实现 计算机工程 2002 年 9 月
【15】金凌紫. 面向对象软件测试技术进展 计算机研究与发展，1998，35(1):6-13
【16】杨芙清，王千祥，梅宏，等 基于复用的软件生产技术 中国科学(E 辑)，2001,31(4):363-371
【17】杨芙清，梅宏，吕建等. 浅谈软件技术发展 电子学报 2002，30(12A)：1901-1906
【18】王言志，刘椿年. 区间算术在软件测试中的应用，软件学报，1998，9(6):438-443
【19】侯勇，张海林. 自动化测试中的关键字驱动脚本技术 电子科技，2006，2:51-54
【20】宫云战，刘海燕，万琳等. 软件测试性的分析与设计技术研究，2000 年全国测试学术会议(CTC'2000) 北京:2000，271－274
【21】陈振强，徐宝文，许蕾，张斌. 一种并发程序可测试性分析框架 计算机学报 2003,12 Vol.26 No.12:1685-1688
【22】IBM Rational 自动化测试技术白皮书

http://www-900.cn.ibm.com/cn/software/rational/resource/software_test.pdf
【23】Elfriede Dustin, Jeff Rashka, John Paul. Automated Software Testing: Introduction, Management, and Performance. Addison Wesley Longman.1999
【24】Krause, Michael H, " A Maturity Model for Automated Software Testing", Medical Device and Diagnostic Industry Magazine, December ,1994
【25】Nagle, Carl, "Test Automation Frameworks", http://safsdev.sourceforge.net/FRAMESDataDrivenTestAutomationFramewors.htm
【26】Pettichord, Bret, "Seven Steps to Test Automation Success", July 16,2000, http://www.io.com/~wazmo/papers/seven_steps.html
【27】Powers, mike, "Style for Making Test Automation Work", Jan 1997, Testers' Network http://veritest.com/tester'snetwork
【28】Strang,Richard, "Data Driven Testing for Client/Server Applications", Conference on Sotware Testing, Analysis and Reliability,pp:395-400
【29】Zambelich, Keith, "Totally Data-Driven Automated Testing", whitepaper, Automated Testing Specialists, http://www.auto-sqa.com/articles.html
【30】E.Dystin, J.Rashka, J.Paul. "Automated Software Testing", Addison Wesly, Reading, MA,1999
【31】Peer "Are you ready for automation testing?" http://www-128.ibm.com/developerworks/cn/rational/r-testauto/
【32】Mike Kelly, Choosing a test automation framework http://www-128.ibm.com/developerworks/rational/library/591.html#N10223, 20 Nov 2003
【33】E J .Weyuker. Testing component-based software: A cautionary tale[J] IEEE Software,1998,15(5):54-59
【34】Y Wu, D Pan, M H Chen. Techniques of maintaining evolving component-based software[C].In Proc of the 16th Int'l Conf on Software Maintenance (ICSM2000).Los .Alamitos, CA:IEEE Computer Society Press, 2000,236-246
【35】Lee S.C , Offutt J., Generating. test cases for XML-based Web component interactions using mutation analysis In:Proceedings of the 12th International Symposium on Software Reliability Engineering, Hong Kong, China 2001 200-209
【36】Sy N.T. Devile Y.. Automatic test data generation for programs with integer and float variables In Proceedings of the 16th Annual International Conference on Automated Software Engineering (ASE 2001) ,San Diego, CA,USA 2001 13-21
【37】Tsai W.T, Chen Y. Paul R , Liao.N, Huang H. Cooperative and group testing in verification of dynamic composite Web services. In Proceedings of the 28th Annual International Computer Software and Applications Conference. Hond Kong, China 2004,170-173
【38】Offutt J. Jin Z., Pan J. The dynamic domain reduction approach to test data generation. Software-Practice and Experience.1999.29(2):167-193
【39】Looker N, Xu J, Assessing the dependability of SOAP RPC-based Web

services by fault injection. In Proceedings of the 9th IEEE International Workshop on Object-Oriented Real Time Dependable Systems Capri Island, Italy 2003,163-170

【40】 Aichernig B.K.. Mutation testing in the refinement calculus Formal Aspects of Computing, 2003,15(2-3):280-295

【41】 Korel B Automated Software Test Data Generation, IEEE Transactions on Software Engineering,1990,16(8):870-879

【42】 Gupta N, Mathur A P, Soffa M L. Automated Test Data Generation using an Iterative Relaxation Method. In: Proceedings of the ACM SIGSOFT 6th Interational Symposium on the Foundations of Software Engineering, Orlando:1998, 231-244

【43】 Korel B. Automated Software Test Data Generation IEEE Transactions on Software Engineering,1990.16(8):870-879

【44】 Gallagher M, Narasimban V L. ADTEST:A Test Data Generation Suite for Ada Software Systems. IEEE Transactions on Software Engineering,1997,23(8):473-484

【45】 Tracey N J A SearchBased Automated Test-Data Generation Framework for Safety-Critical Software:[PhD thesis]. Department of Computer Science ,Univeristy of York, 2000

【46】 Robert C. Martin, "The Test Bus Imperative: Architecture That Support Automated Acceptance Testing", IEEE Software, July/August 2005

【47】 Fevzi Belli, Christof J.Budnik ."Towards self-testing of component-based software", Proceedings of the 29th Annual International Computer Software and Applications Conference 2005

【48】 Wee Kheng , Leow Siau Cheng Khoo, "Automated generation of test programs from closed specifications of classs and test case", Proceeding of the 26th international conference on software engineering 2004

【49】 Atif Meon Adithya Nagarajan, Qing Xie, "Automating regression testing for evolving GUI software", journal of software maintenance and evolution: research and practice 2005

【50】 E.Dsystin, J.Rashka,J.Paul," Automated Software Testing", Addison Wesley, Reading MA,1999

【51】 Michael Kelly Choosing a test automation framework ,Software Engineer, QA,Liberty ,Mutual 20 Nov 2003

【52】 Juichi Takahashi, Yoshiaki Kakuda, "Effective automated testing: a solution of graphical object verification", Proceedings of 11th Asian Test Symposium 2002

在读期间论文发表情况

1 庞晓剑,毛明志. 一种基于状态的面向对象类测试策略,电子产品可靠性与环境试验,2006.06, 8-11.

在读期间参与项目情况

1. 广东省软件技术关键领域重点突破项目：软件开发质量管理提升系统
2. 广东省公路、水路交通信息化“十一五”规划

致 谢

经过一段时间的精心准备，本文从 2006 年 11 月份正式动笔，在撰写过程中，在导师毛明志副教授的指导下，经过多次反复修改，终于得以完成。

首先要衷心感谢我的导师毛明志副教授，导师渊博的学识、敏锐的洞察力与勇于创新的科研精神，严谨的治学态度和对实践的重视，给我很深的启发。导师的言传身教，给我很深的体会，对我以后工作有很好的指导。

感谢姜云飞教授两年以来一直对我的关心并为我排忧解难！感谢中山大学软件技术实验室各位老师同学多年来给予我的帮助与支持。感谢我的家人多年来对我学业的一贯支持。

原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：

日期：2007年5月7日