四 川 大 学

硕 士 学 位 论 文

题　目 ATC 系统软件自动化单元测试工具的研究与实现

作　者 张秀琼　　完成日期 2006 年 4 月 26 日

培 养 单 位 四 川 大 学

指 导 教 师 洪玫　副教授

专　　业 计算机软件与理论

研 究 方 向 软件工程与工具

授予学位日期　　年　　月　　日

ATC 系统软件自动化单元测试工具的研究与实现

计算机软件与理论专业

研究生　张秀琼　　指导教师　洪玫

软件测试作为软件生命周期中不可缺少的组成部分对提高软件质量起着重要作用。随着软件测试技术的发展，自动化测试技术也得到了很大提高。人们在自动化软件测试方面做了大量工作，许多软件公司开发出商业化的软件测试工具，用于实现自动测试。本文介绍了软件自动化测试的概念、技术和发展趋势，总结了当今主流商业化软件自动测试工具的特点，指出了这些工具在技术和功能上存在的不足：这些工具常通过硬编码数据产生测试脚本，不能自动产生测试数据，这可能会导致测试脚本的可重用性和可维护性较差，即当工程发生改变时就无法执行原来的测试脚本。本文分析了 ATC 系统软件测试的现状和在 ATC 系统软件测试中实施自动化测试的必要性，作者在对 ATC 系统软件测试研究的基础上，自主设计并用 VC＋＋V6 实现了一个简单、实用的测试工具 AutomatedTest，用于对 C＋＋程序实现自动化单元测试。

与现有自动化软件测试工具相比，AutomatedTest 工具的特点主要包括：

（1） 通过提取被测程序中的信息来自动产生测试数据，并存入 XML 文档和 MS Excel 工作表中，将测试数据和测试脚本进行分离，可以很方便地对测试数据进行编辑；

（2） 寻找被测程序中所有的方法，自动生成桩函数和驱动函数，进一步实现测试的自动化；

（3） 采用基于数据驱动的自动化单元测试方法，亦即数据驱动脚本技术，将测试输入和期望输出单独存储在 XML 和 EXCEL 表中，

方便修改测试数据，并在不经过修改任何脚本代码的情况下对一个方法进行重复测试。对于测试数据值变化范围较大的方法的测试，此种方法的效率大大高于把测试数据值硬编码到测试脚本中进行测试的效率。

在开发 AutomatedTest 工具的过程中，作者所做的主要工作包括：

（1） 对 ATC 系统的软件测试进行了研究；

（2） 研究了基于数据驱动的自动化软件测试方法；

（3） 采用语法分析的开源代码 ANTLR 来构建 C++分析器，实现了对被测程序信息的提取；

（4） 通过在 VC++中使用 XML 和 EXCEL 组件，实现了用 XML 文档和 EXCEL 工作表来进行测试数据的存取；

（5） 使用宏来实现由存储在 EXCEL 工作表中的数据自动生成被测程序的桩函数和驱动函数，从而进一步实现了测试自动化；

（6） 实现了自动测试验证和将测试结果生成测试报表。

关键词：ATC 系统 自动化软件测试 自动化测试工具 AutomatedTest 工具

The Research and Implementation of An Automated Software Unit Test Tool in ATC System

Major: Computer Software and Theory

Graduate: Zhang Xiuqiong **Supervisor:** Hong Mei

Software testing which is an indispensable part in the life cycle of software plays an important role in improving the quality of software. With the development of software test technology, there is a great advance in the technology of automated software test. Many works have been done in the process of automated software test. Many commercial software test tools were developed for automated testing by software companies. This paper introduced the concept, technique and development trend of automated software test. It also summarized the characteristics of commercial automated software testing tools and pointed out the disadvantage of these tools on techniques and functions. These tools often created testing scripts by hard coding data and can't generate the testing data automatically so that the testing scripts can not be reused and maintained. That means when the project was changed, the original testing scripts can not run efficiently. This paper also analyzed the currently situation of software test in the ATC system and the necessity of automated software test in ATC system. Based on the research of the ATC system software test , author implemented a simple and practical test tool—AutomatedTest—with VC++V6 for the automated software unit test of C++ programs.

Compared with the existing automated software test tools, the characteristics of

the AutomatedTest tool mainly are following.

The first, testing data was generated automatically by way of extracting the information from the tested programs and was saved into the XML files and MS Excel worksheets which can separate the testing data from the scripts so that we can easily edit the testing data.

The second, the tool searched for all methods of the tested programs to create the stub functions and drive functions. Then the automated test was running.

The third, based on the technique of data drive the script , the tool saved the imports of testing and expectant output into the XML files and Excel worksheets so that we can edit the testing data conveniently and test a method for many times with same script code and different testing data. As far as the tested methods with the value changing in a bigger range this approach was more efficient than the approach which runs the testing script with hard coding data.

Within the development of the AutomatedTest tool, the work that the author has done included following:

The first , researched the software test for ATC system.

The second, studied the test principle of the automated unit test tool based on data driving;

The third, realized the function of extracting the information of the tested programs adopting the open source code parser of ANTLR to create the C++ parser.

The fourth, realized the access of the testing data in the XML files and Excel worksheets by use of COM of the XML and EXCEL in VC++ programming.

The fifth, generated the stub and drive function of the tested programs automatically by the defining Macro in VC++ programming.

The sixth, carried out the verification of automatic testing and realized the function of the test result reporting.

KEYWORDS: ATC system automated software testing automated software testing tool AutomatedTest tool

第一章　绪论

1.1 软件测试概述

软件测试最早可以追溯到软件开发的初期。随着计算机的诞生，就开始了软件开发和软件测试。1983 年，IEEE 提出了软件工程标准术语，软件测试定义为："使用人工和自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或是弄清预期结果与实际结果之间的差别。"该定义明确提出了软件测试以检验是否满足需求为目标。

软件测试是软件质量保证的关键步骤。无论是 ISO9000 的质量体系认证，还是 CMU/SEI 的 CMM 或 CMMI 认证，其中均涉及到测试，因此，不管从何种角度讲，软件测试是一个必不可少的活动，是对软件需求分析、设计规约和编码的最终复审；是软件质量保证的关键步骤。软件测试是根据软件开发各阶段的规约和软件的内部结构，精心设计一批测试用例（包括输入数据及其预期的输出结果），并利用这些测试用例去运行程序，以发现软件中不符合质量特性要求（即缺陷或错误）的过程。目前，许多软件开发机构将研制力量的 40%以上投入到软件测试之中，体现了充分重视软件质量要求。众所周知，软件中存在的缺陷甚至是错误，如果遗留到软件交付投入运行之时，终将会暴露出来。但到那时，不仅改正这些缺陷所花的代价更高，而且往往造成恶劣的后果。由此可知软件测试的重要性。

就软件测试的目的，Grenford.J.Myers 提出了如下观点[1]：

测试是程序的执行过程，目的在于发现缺陷；

一个好的测试用例在于能发现至今未发现的缺陷；

一个成功的测试是发现了至今未发现的多个缺陷的测试。

根据这些测试目的和目标，郑人杰等给出了软件测试准则[2][3]。

软件测试方法按照不同的分类原则可以有多种，文献[4]中有详细的描述。但传统的软件测试活动都可以划到黑盒测试和白盒测试两种方法中[5](亦即功

能性测试和结构性测试[6])：

黑盒测试法又称为功能测试，是把程序看作一个黑盒子，完全不考虑程序的内部结构和处理过程，只是根据程序功能说明来设计测试用例。也就是说，黑盒测试是在程序接口进行的测试，它只检查程序功能是否能按照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并产生正确的输出信息，程序运行过程中能否保持外部信息的完整性。一般黑盒测试有如下几种方法：等价分类法、边界值分析法、因果图法、错误推测法。这几种方法没有一种方法能提供一组完整的测试用例，因此，需要把各种方法结合起来使用。

白盒测试法又称为结构测试，与黑盒测试法相反，它的前提是可以把程序看成装在一个透明的白盒子里，测试者完全知道程序的结构和处理算法。这种方法按照程序内部的逻辑测试程序，检测程序中的主要执行通路是否都能按预定要求正确工作。即白盒测试是根据程序的内部逻辑而设计测试用例，其宗旨就是测试用例尽可能提高程序内部逻辑的覆盖程度，最彻底的白盒测试是能够覆盖程序中的每一条路径。因此软件企业需要建立一些标准，即：语句覆盖率、分支覆盖率、条件覆盖率、分支/条件覆盖率、条件组合覆盖率。一般要求100%语句覆盖率、80%～90%的分支覆盖率、60%～80%的条件覆盖率。

现代软件工程中，软件测试与软件开发过程都是平行进行的，软件测试始终贯穿于整个软件开发过程，在软件开发过程中，软件测试进行得越早越好。早期的测试参与不仅确保有效的测试设计，而且能够早期检测错误并且避免错误由需求规格说明转移到设计，再从设计进入代码。这种避免措施降低了成本，减少了返工并节省了时间。在开发周期中，错误被发现得越早，它们则容易被纠正且成本低[7]。软件开发过程中的测试主要包括：

1、单元测试

单元测试是完成对软件设计的最小单位正确性检验的测试工作。单元测试主要依据是软件详细设计文档，其目的是发现在程序单元内部所有重要的控制路径中可能存在的各种错误。单元测试需要从程序的内部结构出发设计测试用例。多个程序单元可以独立、平行地进行单元测试。单元测试一般采用白盒测试方法，辅之以黑盒测试方法。其中包括：逻辑覆盖、语句覆盖、判定覆盖、条件覆盖、判定－条件覆盖、条件组合覆盖、路径覆盖等内容[4]。

由于软件单元往往并不是一个可以独立运行的程序单元，在单元测试的时候，要同时考虑与外界的联系，这就需要用一些辅助模块，即驱动模块（driver）和桩模块（stub）[1]，来模拟与被测程序单元有联系的其他部分。在绝大多数应用中，驱动模块只是一个接收测试数据，并把数据传送给（要测试的）模块，然后打印相关结果的“主程序”。毫无错误的程序“桩模块”的功能是替代那些隶属于本模块（被调用）的模块，这种程序桩模块可能要使用子模块的接口，才能做一些少量的数据操作，并验证打印入口处的信息，然后返回[5]。

2、集成测试

集成测试也称作软件组装测试，通常是在单元测试的基础上，把程序的基本单元按照软件结构设计要求，组装成软件系统。根据软件项目规模的大小，可以将软件的集成测试细分为：部件（或构件）测试和配置项（或子系统）测试，逐步组装成一个可以运行的软件系统。

软件集成测试一般采用黑盒测试方法，辅之以白盒测试方法。其中包括：等价类划分、边界值分析、错误推测方法、因果图等。集成测试方式有非增量组装测试方式和增量式组装测试方式。后者又有自顶向下和自底向上两种方法。其中，自顶向下组装测试方式的优点是能够较早地发现在主要控制方面的问题，缺点是需要建立桩模块，并且随着组装层次的向下移动，桩模块逐渐增加。而自底向上组装测试方式的优点不需要桩模块，而建立驱动模块一般比建立桩模块容易，并且随着组装层次的向上移动，驱动模块逐渐减少；同时由于涉及到复杂算法和真正输入/输出的模块最先得到组装和测试，可以把这些容易出问题的部分在早期解决；此外自底向上组装测试方式可以实施多个模块的并行测试，以提高测试效率。其缺点是对主要控制直到最后才接触到。

3、系统测试

软件系统测试是基于一定的计算机硬件环境，对整个软件进行的一系列测试。它应根据软件项目系统级的有关文档（如系统设计文档、软件开发任务书、软件开发技术合同、软件需求规格说明等），开展软件系统测试工作。系统测试是将已经通过集成测试的软件与具有一定代表性的计算机实用环境相结合，主要检查软件系统自身存在的错误和缺陷，检查软件与系统定义不符合或与之矛盾的错误，检查软件与需求的符合性，检验并确认软件在整个系统中功能、性

能的正确性。主要采用黑盒测试方法。

4、验收测试和配置审计

软件验收测试可以是以用户为主的测试。一般，在软件系统测试结束以及软件配置审查之后，可以开始软件系统的验收测试。验收测试应由用户、测试人员、软件开发人员和质量保证人员共同组成测试小组，设计测试用例。使用真实数据作为输入测试数据，分析检查测试输出的结果，验证软件系统是否达到用户要求等。

软件开发的 V 模型说明了何时应进行测试[9]。V 模型指出每个开发活动都有相应的测试活动。每一层的测试检验都对应于相应的开发活动。无论使用哪种软件生命周期模型，这一模型都适应。图 1.1.1 所示的简单 V 模型有四个开发和测试活动层次。

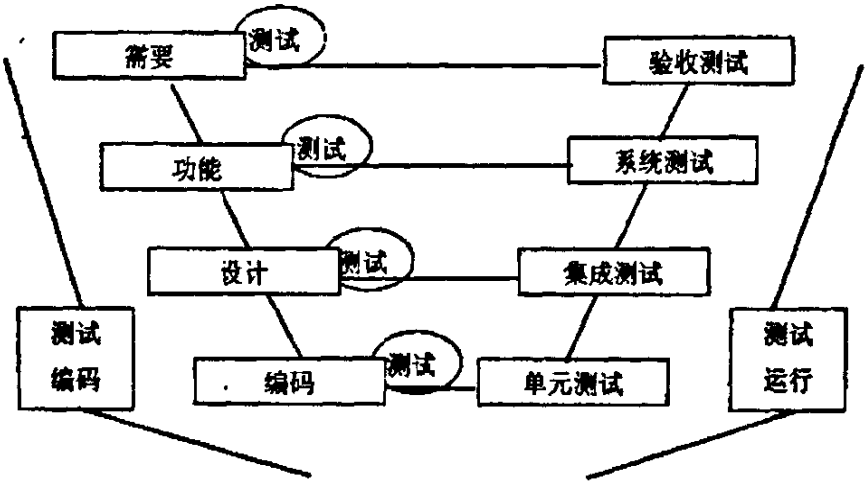


图 1.1.1 早期的测试设计 V 模型

软件测试的执行可用自动执行和手工执行，随着软件规模的扩大，手工测试在效率上和开销上都显出了其局限性，而自动化测试可以显著减少测试开销，同时显著增加在有限时间内的测试，由此成为了软件测试技术研究的热点，也指引着软件测试的方向。

1.2 测试自动化概述

软件测试是一项艰苦的工作，需要投入大量的时间和精力，据统计，软件测试会占用整个开发时间的40%。一些可靠性要求非常高的软件，测试时间甚至占到总开发时间的60%。但是软件测试具有一定的重复性，我们知道，软件在发布之前要进行几轮测试。在测试后期所进行的回归测试中大部分测试工作是重复的，回归测试就是要验证已经实现的大部分功能，这种情况下，只是为了解决软件缺陷、需求变化，代码修改很少，针对代码变化所做的测试相对比较少。而为了覆盖代码改动所造成的影响需要进行大量的测试，虽然这种测试找到软件缺陷的可能性小，效率比较低，但又是必要的。此后，软件不断升级，所要做的测试重复性也很高，所有这些因素驱动着软件自动化测试的产生和发展。

1.2.1 测试自动化概念

软件测试自动化是相对手工测试而存在的，主要是通过所开发的软件测试工具、脚本等来实现，具有良好的可操作性、可重复性和高效率等特点。自动测试的一般定义为：各种测试活动的管理与实施，包括测试脚本的开发与执行，以便使用一种自动测试工具来验证测试需求[8]。自动化测试是一种测试技术，是软件测试中提高测试效率、覆盖率和可靠性的重要测试手段。

1.2.2 测试自动化的意义

软件测试自动化的意义存在于两个方面：一是手工测试的局限性；二是软件自动化测试所带来的好处。

测试人员在进行手工测试时，具有创造性，可以举一反三，从一个测试用例想到另外一些测试用例，特别是可以考虑到测试用例不能覆盖的一些特殊的

或边界的情况。同时，对于那些复杂的逻辑判断、界面是否友好，手工测试具有明显的优势，但是手工测试在某些测试方面，还存在着一些局限性，包括：

1、通过手工测试无法做到覆盖所有代码路径。

2、简单的功能测试用例在每一轮测试中都不能少，而且具有一定的机械性、重复性，其工作量往往较大，却无法体现手工测试的优越性。

3、许多与时序、死锁、资源冲突、多线程等有关的错误通过手工测试很难捕捉到。

4、在系统负载、性能测试时，需要模拟大量数据或大量并发用户等各种应用场合时，也很难通过手工测试来进行。

5、在进行系统可靠性测试时，需要模拟系统运行 10 年、几十年，以验证系统能否稳定运行，这也是手工测试无法模拟的。

6、如果有大量（几千）的测试用例，需要在短时间（1 天）内完成，手工测试几乎不可能做到。

7、测试可以发现错误，并不能表明程序的正确性。因为不论黑盒、白盒方式都不能实现穷举测试。对一些关键程序，如导弹发射软件，则需要考虑利用数学归纳法或谓词演算等进行证明。

由于手工测试的局限性，软件测试借助测试工具极为必要，并向软件测试全面自动化方向发展，将测试工具和软件测试自动化综合起来，可以解决上述局限性，并且会带来一些好处：

1、缩短软件开发测试周期。软件测试具有速度高、效率高的特点，对上千个测试用例，软件测试自动化工具可以在很短时间内完成，还可以在很短时间内运行同样测试用例 10、100 遍。

2、测试效率高，充分利用硬件资源。可以在运行某个测试工具的同时运行另一测试工具，也可以在一面运行某个测试工具一面思考新的测试方法或设计新的测试用例，能够把大量测试个案分配到各台机器上同时运行，从而节省大量的时间。也可以把测试安排在晚上及周末运行，这样能提高效率。

3、节省人力资源，降低测试成本。在回归测试时，如果是手工方式，就需要大量的人力去验证大量稳定的旧功能，而通过测试脚本和测试工具，只要一个人就可以了，可以节省大量的人力资源。同样的测试用例，需要在很多不同

的测试环境下运行，这也正是测试工具大展身手的时候。

4、增强测试的稳定性和可靠性，通过测试工具运行测试脚本，能保证100%进行。

5、提高软件测试的准确度和精确度。软件测试自动化的结果都是数量化，能够同所预期结果或规格说明书规定的标准进行量化对比。

6、软件测试工具使测试工作相对比较容易，且能产生更高质量的测试结果。

7、手工不能做的事情，软件测试自动化能做，如负载、性能测试。

软件测试实行自动化进程，决不是因为厌烦了重复的测试工作，而是因为测试工作的需要。

1.2.3 测试自动化的原理和方法

软件测试自动化实现的基础是可以通过设计的特殊程序模拟测试人员对计算机的操作过程、操作行为，或者类似于编译系统那样对计算机程序进行检查。软件测试自动化的原理和方法主要有：直接对代码进行静态和动态分析、测试过程的捕获和回放、测试脚本技术、虚拟用户技术和测试管理技术。

1.2.4 测试自动化工具

自动化测试需要测试软件的支撑，一般来说，这些测试软件是由经验丰富的测试人员精心设计的，在测试软件的基础上再应用自动化技术实现自动测试。在软件开发周期中，不同阶段的测试需要不同测试工具的支持。图1.2.4.1表示了不同测试工具在软件生存周期中的位置。

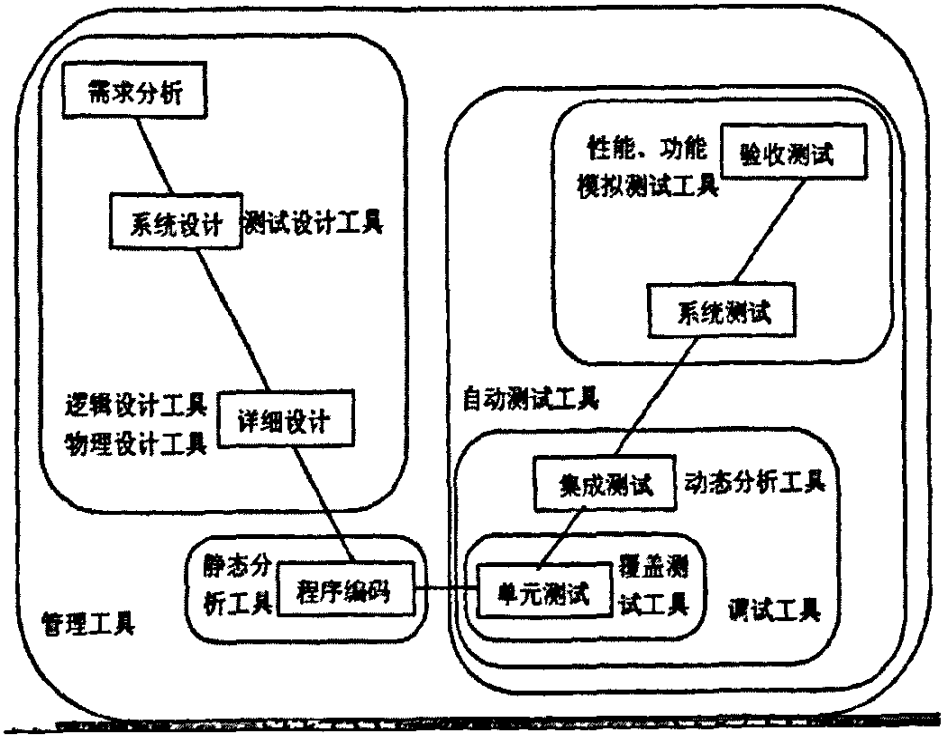


图 1.2.4.1 软件开发生命周期中适用的测试工具

测试工具根据测试方法的不同，主要分为白盒测试工具和黑盒测试工具。

白盒测试工具是针对程序代码、程序结构、对象属性、类层次等进行测试，测试中发现的缺陷可以定位到代码行、对象或变量级。单元测试工具多属于白盒测试工具。

黑盒测试工具是利用脚本的录制/回放，模拟用户的操作，然后将被测系统的输出记录下来同预先给定的标准结果比较。适用于系统功能测试和性能测试，包括功能测试工具、负载测试工具、性能测试工具等。

1.2.5 测试自动化的误区

测试自动化使用过程中可能会遇到许多问题，以下是普遍存在的测试自动化误区[10]：

1、不现实的期望。测试并不能解决所有问题，测试自动化也不例外，对于测试自动化不应该赋予不现实的期望值。

2、期望自动测试发现大量新故障。测试在首次运行时最有可能发现故障，如果测试已经运行并通过，再运行相同的测试发现新故障的可能性会小很多，除非测试正执行一段已修改过的代码或由于软件其他部分的修改影响到该代码，或在不同环境下运行。测试自动化主要用于重复已经运行过的测试，而不是用来发现大量新的故障。

3、安全性错觉。自动测试没有发现任何故障并不意味着软件没有故障，因为所进行的自动测试可能不全面，或自动测试本身就有故障。

4、自动测试的维护开销。和普通软件一样，自动测试软件同样需要进行维护，一旦测试软件的维护比手工重新测试更费劲，测试自动化将不具备实用价值。

5、技术问题。如果被测试软件在设计和实现时没有考虑可测试性，则测试时无论是自动测试还是手工测试，难度都会很大，对这种软件，测试自动化不一定是最好的选择。

1.2.6 测试自动化的发展

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实现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。

软件测试技术的自动化是软件测试的发展趋势，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。使用测试自动化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，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与预期输出结果来实现。

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。尤其是对于需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，从而软件自动测试技术在软件工程中的应用越来越广泛。

1.3 国内外研究现状

软件自动化测试的发展大致经历了 3 个阶段。第 1 个阶段是基本的 GUI 测试，这种测试检验了标准的 GUI 对象和控件。测试脚本是非结构化的，并且不可维护。在第 2 个阶段中，脚本编写者发展了“建立结构良好、健壮的、可维护测试”的能力，其关键特征是测试脚本组件的可重用性。测试自动化的第 3 个阶段的特征是控制了测试资源。在这个级别上，测试设计和测试自动化被看成是相互分开的行为。

对于传统的自动化测试，即其发展的前 2 个阶段，当遇到数据输入改变、程序流改变和管理应用程序改变等情况时，由于被改编的数据都是经过硬编码的，若要改变或修改它们，必须编辑或重新记录脚本，这必将消耗大量的时间，达不到自动化测试应有的效果。传统自动化测试失败的主要原因是与捕获/回放方法相关联的维护的负担。而第 3 个阶段的用户却用一个不同的方式使用脚本，即单个脚本处理每个测试用例。由此可看出，软件自动化测试的方向是朝着怎样实现自动化方向发展的，以便进一步减少手工的重复劳动，使测试效率提高。

自动化软件测试一般都是在工具的基础上实现的，国外已有许多专门的商用软件测试工具。现普遍使用的主流商业自动测试工具有：

IBM 的 Rational 系列工具[11]，用于软件开发整个生命周期的工具，包括需

求管理、可视化建模、测试、配置和变更控制。Rational Suite TestStudio 主要提供快速提升功能、可靠性、性能和测试管理等综合测试能力的解决方案。提供了一种集成测试解决方案，使测试人员就产品的功能性、可靠性和性能，进行全方位的质量测试。用此工具，可以方便地解决以下功能测试问题：回归测试、利用数据池方便地解决大批量数据驱动的功能测试、完善的功能测试管理平台。

Mercury 公司的 Mercury Interactive[12]可以测试特制应用程序、打包和应用程序和 Internet 应用程序。该工具套件包括下面的组件：TestDirector，提供了测试计划、测试执行和错误跟踪等功能，按层次文件夹的形式组织测试，从而帮助计划和跟踪测试，跟踪测试包括跟踪人工和自动测试；WinRunner，是 Mercury 的捕获/回放工具，用于测试 Windows 应用程序、Web 应用程序和通过终端竞争访问的应用程序，WinRunner 有许多先进的功能，例如从文件读数据，排除测试运行中的意外出现在屏幕上的窗口干扰；XRunner，是 Mercury 应用于 UNIX 应用程序的捕获/回放工具；LoadRunner，是一个对 Windows 和 UNIX 环境的负载测试工具，该工具能够模拟数千个用户，产生用于评价和测试不同组件性能的场景，例如服务器、数据库、网络组件，该工具提供了定位系统瓶颈的详细报告。

Compuware 公司的 DevPartner Studio[13]具有自动检测、诊断和帮助解决软件错误和性能问题的功能。包括的组件有用于检测 Visual C++应用程序的内存和资源泄漏的 BoundsChecker，应用于对 Java 应用程序进行线程和事件分析的 JChcek 等。另外，NuMega TrueCoverage 用于对 Visual Basic，Visual C++和 Java 应用程序，报告测试脚本已执行了多少代码，是否正在测试不同部分的程序代码，从而确保测试脚本确实对需要测试的代码进行了测试。

Parasoft 公司的 Insure++[14]：是一个检测 C/C++运行时错误的自动测试工具。

ObjectSoftware 公司的 ObjectTester[15]：是一个为处于实现阶段的单元测试产生测试脚本的工具，这个工具把单元测试定义为软件单元的最底层测试。

Segue Software 公司的工具：其 Silk 家族工具[16]是电子商务测试产品，该工具对一些测试阶段实现了自动化，包括单元和回归测试(SilkTest)、负载和性能测试(SilkPerformer)、场景测试(SilkRealizer)。

Software Research 公司的 TestWorks[17]工具：是一个测试工具集成套件，依

据覆盖分析管理测试过程。

另外，还有一些开放测试工具，如：

Ant ，允许开发者以 XML 构建脚本编写测试，这个脚本使用 Java 类实现测试。

JUint，是一个常用于 Java 单元测试的工具，是一个测试框架，允许开发人员通过规定代码的工作方式，运行可重复测试。同时它也是一个回归测试框架。

JProbe Suite，是一个专为 Java 开发者设计的集成套件，具有性能评测、内存调试、代码覆盖、线程分析功能。

HttpUnit，主要用于 Web 应用程序的功能测试，或黑盒测试。

还有很多测试工具就不一一列举了。

但在国内，由于软件质量的重视程度还不够，有很多软件公司对自己的产品测试都没有做到系统化和规范化，自动化的软件测试工具的开发也很少。不过，随着软件企业各方面的不断强大和发展，软件质量的进一步被重视，将会有很多适合于本企业的测试工具被研发出来。这也是自动化测试在现今成为研究热点的一个原因。

1.4 课题来源

本课题来源于空中交通管制系统（简称 ATC 系统）建设系列项目。ATC 系统是新一代具有较高自动化程度的空中交通管制系统，通过自动化的日常作业，完成在机场管制区域范围内对所有的飞行活动进行监视、组织实施飞行管制等各种空中交通管制任务。它是一典型的实时系统，它具有及时接收和处理雷达数据、及时接收和发送飞行情报、及时进行空中交通管制等功能。

ATC 系统属于长期不间断运行的专用、使命重大、多任务的实时系统，因此要求 ATC 系统必须具有高安全性和高可靠性。在当前的实时软件开发中采用的测试技术仍旧为传统的结构化测试技术，如何使用自动化测试技术来提高测试效率、减少测试开销，这在 ATC 系统的测试中是很值得研究的问题，同时也具有非常重要的实用性。基于此，本文重点探讨了应用于 ATC 系统软件单元测

试的自动化测试工具 AutomatedTest 的开发及功能。

1.5 研究目的和意义

目前普遍使用的主流商业自动测试工具许多都覆盖了软件测试生命周期的各个阶段。但是，在产生测试脚本或程序方面都采用了类似方式，即人工指定被测产品，指定被测方法，编辑和调试产生的测试脚本。并且目前还缺乏一种明确标准化的测试技术用于软件测试，所以导致了测试工具的如下一些弱点：

1、缺乏彻底测试被测产品的能力

2、缺乏实现集成和互用测试的能力

3、缺乏自动产生测试脚本的机制

4、缺乏严格的测试，用于决策何时可以可靠地终止测试

5、缺乏性能度量和测试度量过程

为了弥补这些不足，需要软件团体自己开发自动软件测试工具。本文在对 ATC 系统软件测试进行了充分的研究基础上，开发了 AutomatedTest 工具，来实现对用 C++编写的程序代码进行自动化单元测试，以此来减少手工劳动的重复性，为 ATC 系统软件的质量和可靠性提供了保障。

采用自动化测试工具，对一个软件企业来说是很重要的，它可以减少在重复劳动中多次的人力投入，并且可以有效地利用一切时间进行测试，从而提高测试的效率。自主开发自动化软件测试工具对一些开发大型项目的企业来说尤为重要，因为商用测试工具不可能满足每一个软件企业的需求，软件团体开发自己的自动软件测试工具是很有必要的。

研究的理论意义：软件测试是软件质量保证的关键，它代表了从系统需求到软件设计、编码、维护全过程的质量检查。对软件测试技术和方法的研究，是进一步完善软件顺利开发全过程的理论基础。

现实意义：对 ATC 系统的测试有利于充分保证该系统的稳定性、成熟性、健壮性，对 ATC 系统在实际的应用中的高安全性和高可靠性具有重大意义。

1.6 作者所做的工作

作者对自动化软件测试做了详细的研究，并结合实习中的项目实践对理论有了近一步的理解。在文章中详细描述了自己所开发的用于实现自动化单元测试的工具AutomatedTest的功能设计及各功能模块的实现。

在实践及本文中作者所做的工作包括：

1、参与了ATC系统后期的软件开发，主要负责一个回放显示程序的开发和维护。在此过程中，作者对项目的软件开发工作有了一定的实践经验。

2、参与了ATC系统各种类型的测试工作，包括：单元测试、集成测试、系统测试、工厂验收测试及用户现场交付验收测试等。从中对ATC系统作了充分的了解，对其结构，各功能模块及其功能都有详细的了解。并且对整个系统的测试过程也有了充分的了解和实践，为进一步对ATC系统软件的测试进行研究提供了实践依据。

3、参与了ATC系列项目部分文档的编写，如小组内程序的部分需求分析文档，设计文档，程序分析文档等。通过这些实践，使作者对整个软件工程有了进一步的了解，同时为自己开发自动化单元测试工具作了准备。

4、在进行了充分实践的基础上，对ATC软件测试和软件测试技术作了详细的研究。由于ATC是一个使命重大的系统，其要求的高质量和高可靠性就决定了对ATC系统的测试必须要非常充分，基于ATC系统现在大多是进行手工测试，所以作者研究了对其进行自动化测试的必要性。对ATC系统进行自动化测试，不仅能提高系统的测试效率，对一个公司来说，还能减少其在测试费用上的开支。

5、作者自主开发了用于实现自动化单元测试的工具AutomatedTest，并在本文中作了详细描述。

在整个AutomatedTest工具的开发过程中，作者作了如下工作：

1、研究了自动化单元测试工具采用的基于数据驱动的自动化软件测试方法的测试原理。

2、采用语法分析的开源代码 ANTLR 来构建 C++分析器，实现了对被测程序信息的提取功能。

3、通过在VC++中使用XML和EXCEL组件，实现了用XML文档和EXCEL工作表来进行测试数据的存取。

4、使用宏来实现由存储在EXCEL工作表中的数据自动生成被测程序的桩函数和驱动函数，从而进一步实现了测试自动化。

5、实现了自动测试验证，并将测试结果生成测试报表的功能。

经过实践证明，所开发的AutomatedTest工具能较方便地实现自动化单元测试。

1.7 本文的章节安排

本文分为六章展开。

第一章介绍了软件测试的一些基本概念，概述了自动化测试的相关理论，及国内外的研究现状，给出了该课题提出的背景，课题研究的意义，简述了作者所作的工作。

第二章对 ATC 系统和 ATC 系统软件测试作了简要介绍。研究了对 ATC 系统软件测试实行自动化测试的必要性以及用 AutomatedTest 工具所能带来的效益。详细描述了 AutomatedTest 工具所采用的基于数据驱动的自动化软件测试方法。

第三章介绍了 AutomatedTest 工具的功能设计。

第四章详细介绍了各功能的实现，并就实现过程中的关键技术进行了研究并给出了解决方案以及编码实现方案。

第五章演示了用 AutomatedTest 实现自动化单元测试的过程及其具有的功能。

第六章总结全文并指出了 AutomatedTest 的不足，阐述了用 AutomatedTest 工具进行自动化测试还需要改进和进一步研究的问题。

第二章 ATC 系统的测试

2.1 ATC 系统简介

ATC 系统是新一代具有较高自动化程度的空中交通管制设备，通过在自动化的日常作业，完成机场管制区域范围内所有飞行活动进行监视、组织实施飞行管制等各种空中交通管制任务。为满足上述要求，机场管制中心系统的建立目标如下：

（1） 实现雷达、ADS 数据接收和处理自动化；

（2） 实现飞行数据接收、传输、处理和管制中心内部分发自动化；

（3） 提高管制工作手段、作业自动化的程度；

（4） 实现管制中心系统间的情报交换自动化；

（5） 实现 ATC 系统全国飞行流量监控中心、技术监控维修网、气象情报综合处理交互系统、空域管制和航行情报系统之间的情报交换和处理自动化；

（6） 提供话音、数据连续记录和同步重演能力；

（7） 提供系统监视和控制能力；

（8） 具备模拟训练、演示和测试的能力；

（9） 具有系统设备与接口扩充能力和兼容性；

（10） 全用适应性的现场配置数据支持系统的可移植性。

ATC 系统具有开放式的系统结构。系统具备扩充能力，可以满足空中交通量的预计增长的需要。在系统预期寿命期内具备升级能力，可以满足操作和技术变化的需要。

2.2 ATC 系统的测试

ATC 系统是一个典型的实时系统，结构复杂，功能多，性能要求高。为了

保证其质量，在整个系统研制过程中都进行了相应的测试。对 ATC 系统主要进行了如下测试：

单元测试，在编写代码时进行。

集成测试，各模块代码编写完，将其集成起来所进行的测试。

系统测试，在编码完成后，对系统进行的测试，包括功能测试、压力测试、性能测试、容量测试等。系统测试主要采用手工测试，即由人工根据事先写好的测试用例来逐条逐个操作地完成。

对 ATC 系统的测试，文献[18]中进行了面向对象的测试研究，文献[19]中对系统性能测试、产品交付测试等提出了用虚拟测试系统的方法。本文则针对 ATC 系统软件测试提出了一种自动化测试的方法，即用所开发的 AutomatedTest 工具实现对 ATC 系统软件的自动化单元测试。

2.3 ATC 系统软件的自动化测试研究

ATC 系统是一个大型实时、任务重大的复杂系统。要保证其高可靠性和高质量，必须经过大量测试。然而，在这些测试中，重复性测试占具 90%以上，为了提高测试的效率，以及增加测试次数，很有必要对其进行采用自动化测试的方法，以此减轻测试人员的工作压力，使其能够把更多的时间和精力投入到高风险区域的问题上。

在现有的对 ATC 系统的测试中，测试人员基本都是进行系统测试，而代码开发初期应进行的单元测试及代码集成阶段所进行的集成测试都是由程序员在进行，主要是通过代码走查以及程序运行中使用日志文件来实现单元测试和集成测试，这样就可能使单元测试进行的不充分，也不能保证测试的覆盖率。这就导致了在系统集成后进行系统测试阶段所发现的缺陷，有时可能是由某个程序代码模块中的一个变量的使用而导致，或者是某个分支没有执行而导致的缺陷，这类缺陷如果在单元测试进行充分的情况下，应该是可以尽早发现的。由此可知，软件的单元测试是测试过程中最基本的测试，单元是软件的构成基础，因此单元的质量是整个软件质量的基础。单元测试是软件测试过程中直接与代

码相关的测试，它对其它测试步骤的进行有重要影响。

针对以上问题，本文提出了一种单元测试自动化方法，以此来提高 ATC 系统单元测试的充分性及测试的覆盖率，从而为提高整个软件的质量奠定了坚实的基础。自动化单元测试的实现由所开发的自动测试工具 AutomatedTest 来完成，主要是采用了一种数据驱动的单元测试自动化方法，通过收集被测程序的信息，自动产生测试用例和测试数据，并用此测试数据驱动测试脚本的产生，来自动完成单元测试，产生测试报告。

自动测试工具 AutomatedTest 其主要特点为：

改善单元测试：现有的测试工具通过逆工程能够产生单元测试脚本，然而，确定测试的单元或方法的过程必须由人工完成，而且这样的一个测试脚本一次仅测试一个方法。当某个类有许多方法时，产生的测试脚本数量也就多，因此导致测试工作量激增，自动测试的优越性不再明显了，不可能实现所有方法的测试。本工具能够自动寻找被测程序中所有的方法，测试者仅需把被测程序提交给工具，再由测试脚本来完成对所有的构造函数、方法和属性的测试，这样来实现对被测程序的彻底测试。

自动产生测试数据：使用现有的工具编写测试用例是一项困难的任务。为了给参数赋合适的值，需要花时间研究被测应用程序的每个模块，这是目前阻碍软件测试自动化的主要问题之一。ATC 系统自动单元测试工具 AutomatedTest 能够基于每个参数的本身特点，自动产生测试数据值，还可以利用开发人员编码时存储到 XML 文档中的测试用例，因此，AutomatedTest 工具能够自动精确产生测试数据，并节省了测试人员的时间，使测试人员可以集中精力解决高风险区域的问题。还可以对产生的测试用例进行编辑和修改。当修改测试用例时，数据存储能够指示有效值，另外，为了测试应用程序的错误处理能力，测试人员可以把无效值赋给参数。一次可以为测试脚本提交多个不同的测试用例，因此，该工具可以在短期内测试许多测试用例和数据路径，从而找到最多的缺陷。

作为一个企业来说，采用任何测试工具最先考虑的问题应该是其能带来的经济效益问题，采用自动化测试工具进行单元测试能带来的效益主要表现如下：

1、保证局部代码的质量。单元测试在隔离的前提下，分别对各个代码单元进行测试，能够达到其他测试不可能达到的测试完整性，从而保证了局部代码

的质量。只有局部代码的质量得到了保证，代码的整体质量才可能得到保证。

2、保证代码的整体结构良好。要对代码进行单元测试，最起码的前提是代码能够隔离，也就是说，要具有一定的可测性，因此，单元测试是一种有效的约束机制，这种机制将保证代码具有良好的整体结构。例如，如果把业务代码直接写在界面类中，将很难进行单元测试，随意的不合理的紧耦合也会造成难于测试，单元测试使这些不好的特性得于及时发现，从而很容易进行修正。可以说，可测性是高质量代码的最重要的特性，不具有可测性，也就无法衡量代码的正确性，而有了可测性，也就在一定程度上保证了代码的可扩展性、可复用性。

3、单元测试使排除代码错误的成本最小化。排除错误的代价随着时间的推移呈指数级数上升，并且，有很多细小的代码错误只有通过单元测试的手段才能发现。

4、单元测试大幅度降低了后期测试和升级维护的时间成本。如果代码经过了充分的单元测试，集成测试和系统测试就只需要关注设计方面的问题。自动回归测试也大量降低升级维护成本。

5、单元测试自然地使开发流程变得“敏捷”，可以适应频繁变动的需求，因为整体结构良好的代码具有较好的可扩展性，自动回归测试又能保证修改不会引入新的错误。

对程序员来说，单元测试也有利于养成编写局部代码时的缜密的思维习惯，以及提高设计能力。

2.4AutomatedTest 的基于数据驱动的自动化单元测试方法

单元测试是开发者必须完成的过程。单元测试应该从开发者的角度去写，并且要集中在被测试类的特定方法上[20]。自动化单元测试是弥补手工单元测试的不足，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。单元测试的任务包括：模块接口测试、模块局部数据结构测试、模块边界条件测试、模块中所有独立执行通路测试、模块的各条错误处理通路测试[21]。测试都是根

据测试用例来进行的，测试用例是用编程语言或更方便的脚本语言写出短小的程序来产生大量的测试输入(包括输入数据与操作指令)，或同时也按一定的逻辑规律产生测试输入[31]。自动化单元测试的执行也是根据测试用例进行的，因此，开发自动化单元测试工具 AutomatedTest 就必须要解决好测试数据的构造、测试脚本的生成等问题，因其直接影响了自动化测试工具在执行时的自动化程度。

一、基于数据驱动的自动化单元测试方法

目前针对单元测试的测试数据产生的研究很多：文献[22]中研究了软件测试数据生成的链接方法，这种方法是在被测程序实际执行的基础上产生测试数据的，运用了数据相关性分析来自动识别语句，指导搜索过程。文献[23]中介绍了软件结构测试数据生成的研究现状，指出当时按路径生成测试数据的方法主要有符号执行和程序直接执行两种。文献[24]中研究了基于域错误的数据生成算法。Gupta 等人提出了一种基于程序执行的方法[25]，使用松弛迭代法生成测试数据。文献[26]中研究了对于包含过程的程序自动生成测试数据。另外还包括基于数据流分析技术的测试数据生成[27]，面向断言的测试数据生成方法[28]，运用约束解决技术生成测试数据的方法[29]，应用区间算术对约束集求解生成测试数据的方法[30]等。

而针对代码进行白盒级的测试系统中，在进行白盒测试过程中都不同程度需要使用具体的编程语言或是测试系统提供的脚本来编写测试所需要的驱动和桩；即使是应用比较成熟的单元测试工具或者自测工具，在测试中也都不可避免的需要构造测试数据，主要包括被测试函数入参和桩函数的返回值等[32]。

在测试数据的构造上，通常有如下方法：

方法一：使用 C 书写测试代码和桩代码，在测试代码中构造测试数据，包括测试函数的入参数据和桩函数的返回值数据，然后和被测代码一起编译运行，实现对函数接口进行测试。

方法二：使用脚本书写测试函数的测试用例，脚本中构造测试数据，然后将脚本用例转换为 C 代码和被测代码一同编译，实现对函数接口的测试。Rational 的 Test RealTime 单元测试工具中对函数接口的测试就是采用这种方法来构造的测试数据。

方法三：使用一种镜像技术[33]，将被测代码中全局变量等数据映射到测试脚本中，实现在脚本中对数据的控制，特点在于脚本解释器内嵌到被测代码，直接将变量镜像为脚本对象，实现对变量数据的操作。

方法一和方法二虽在构造桩和驱动的接口上存在差别，在数据驱动的构造上是相同的，都需要有构造测试数据的代码，并和被测代码一同编译，实现对函数接口的测试。方法三提供的镜像技术，通过驻留在被测代码中的脚本解释器，实现对结构数据的脚本访问接口，其对数据的操作还是和被测代码联系在一起，同时在测试脚本中构造数据。

从上分析可以得出在单元测试中数据的构造都是通过和被测代码编译在一起的“辅助代码”来实现，使得单元测试中数据驱动方式的白盒测试的数据构造和结果数据的分析效率比较低，方法二和方法三使用脚本来构造测试用例，不能做到脚本和数据的分离，这样导致测试数据的变化会同时带来测试代码(或是脚本)的变化。

针对以上不足，AutomatedTest 提出了一种基于数据驱动方式的单元测试自动化系统和方法。使用此测试工具通过一次提取被测代码中的结构信息，即可在后续测试中高效的进行测试数据的构造和测试结果分析，测试数据的构造和测试结果分析不再通过测试代码在编译时引用被测代码中的相应结构定义信息的方式来实现，简化了测试代码(或是测试脚本)的书写。

另外通过测试代码和数据的分离做到以测试数据驱动测试代码或是脚本的运行来实现白盒测试，后续针对已有的测试用例添加新的测试数据或是修改测试数据即可，测试代码不需任何改动，避免因测试数据和测试代码绑定在一起而导致的测试代码的频繁改动和重复编译。

二、测试数据的构造

在 AutomatedTest 中，构造测试数据的步骤如下：

步骤 1：规范被测程序，按测试工具的要求来格式化程序。也就是直接将测试用例作为注释的内容在程序中标出。并给出默认初始值信息和具体参数的值(也可以不给)。

步骤 2：用测试工具提取被测代码中的信息，生成自定义形式的描述信息，用 XML[34]进行存储。

步骤 3：在具体使用这些数据之前，可以先修改或编辑具体的参数数值。然后再用脚本进行测试。

在实际应用中，如果一个项目决定采用工具进行测试的话，那第一步由程序员就能很轻松地完成。由程序员编写单元测试用例是再合适不过，并且现在的单元测试也基本都采用此办法，因为程序员最了解其编写的代码应该测试什么，这样可以大大提高测试工具的测试彻底性。如果程序员都遵守测试工具的约定，那第一步就可省略了。

此种测试数据构造方法具有如下优点：

与方法一相比，测试中测试数据的构造不再以通过写 C 代码实现测试数据的构造，采用提取被测试代码中构造的测试用例信息来构造测试数据，为用户编辑和构造测试数据提供了更灵活更方便的方式。

与方法二相比，方法二需要在脚本中构造数据，脚本和数据绑定在一起，并生成相应的源码；和被测代码一同编译，完成对函数接口的测试，在后续测试中若对某个接口添加新的测试数据或是修改测试数据，需重新编写或是修改脚本，生成相应的测试代码，和被测代码重新编译，不可避免进行重复编译。

AutomatedTest 中测试数据的构造采用提取被测程序中的相关信息来生成，做到了脚本和数据分离，脚本只用于测试的控制；数据构造方式提供了简单方便的数据编辑方式。

与方法三相比，方法三借助镜像技术，对数据的操作和构造是在脚本中进行，脚本和数据绑定在一起，在增加测试数据时不可避免书写新的测试脚本或是修改脚本。AutomatedTest 做到了脚本数据分离，在针对接口进行测试时，数据的增加或修改能避免书写新的测试用例或是修改脚本。

三、测试脚本的生成

非脚本测试是指坐在计算机前边测试边想测试什么，没有测试计划也没有测试列表。这往往在规模较少、时间紧迫，甚至没有按照软件工程要求进行开发时采用。显然不能适用于软件自动测试。模糊的手工脚本，它含有测试用例的描述，但不对输入和比较进行详细描述，测试条件可能是隐含的而不是显式说明的。模糊的手工脚本用于自动测试，取决于实施者的技能，要求他能确定测试条件且对软件和应用有所了解。详细的手工脚本，它包含准确的测试输入

数据和相应的测试输出结果，可以严格按脚本进行测试。在此基础上进行自动测试就比较容易。自动脚本，它包含测试工具中使用的数据和指令。如同步、比较信息、从何处读数据和将数据存放在何处，以及控制信息。下面只讨论自动脚本。可维护的脚本技术非常类似于建立可维护的程序，所以脚本技术类似于编程技术，脚本是由脚本语言编写，而脚本语言就是一种编程语言。因此它应遵循的原则是：注释：为用户和管理者提供帮助；功能：执行单个任务且可复用；结构：应易读、易理解和易维护；文档：有助于复用和维护。

脚本技术有如下几种：

线性脚本：是在录制手工执行测试用例时得到的脚本。因此每个测试用例可以通过脚本完整地被回放。

结构化脚本：类似于结构化程序，它含有 3 种基本控制结构（其“顺序”结构就是线性脚本），但还可以有“调用”。因此不仅提高了复用性，也增加了功能和灵活性。

共享脚本：可以被多个测试用例复用，即脚本语言允许一个脚本被另一个脚本调用，因而它可以放在一个地方而不必放在每个脚本中。这样能减少维护开销，但应在稳定性上花费更多精力。如果要充分发挥共享脚本的优点，则好的技术人员和配置管理系统十分重要。

数据驱动脚本：将测试输入和期望输出储存在独立的（数据）文件中，而不是存放在脚本中。这样只需修改数据表便容易增加新测试而无需修改脚本。注意脚本和数据表必须一致。

关键字驱动脚本：实际上是较复杂的数据驱动的逻辑扩展。将关键字储存在数据表中，用关键字指定可执行的任务。控制脚本可以解释关键字，这就要求一个附加的技术实现层。应正确表示关键字。这种脚本不需要像其他脚本技术那样说明细节，只需告诉测试做什么，真正做到测试信息与实现的分离。

AutomatedTest 采用数据驱动脚本技术，即将测试输入和期望输出单独存储在 XML 和 EXCEL 表中，这样可以很方便的修改测试数据，在不经过修改任何脚本代码的情况下可以多次对一方法进行重复测试。对于测试数据值多变化，域范围较大的方法的测试，此种方法的效率就得到了很好的体现，其效率大大高于把测试数据值硬编码到测试脚本中进行测试的效率。这种方法在.NET 中已

得到了很好的应用[37]。

AutomatedTest 采用数据驱动方式进行白盒测试，使得测试数据的构造通过对被测代码中结构信息提取和基本数据类型的特殊处理，做到测试数据的构造在后续测试中独立于被测代码；通过脚本和数据分离简化了脚本的书写的同时避免了因测试数据和脚本绑定在一起而导致的测试脚本的频繁改动和重复编译；在 AutomatedTest 的单元测试自动化系统中驱动和桩代码相对单一和简练，在具体实施中可以自动生成被测系统中的驱动和桩代码，使用 AutomatedTest 进行自动化单元测试可以大大提高白盒测试的测试效率。

2.5 本章小结

本章对 ATC 及 ATC 系统测试作了简单介绍，分析了 ATC 单元测试中存在的不足，提出了一种自动化单元测试的方法。并介绍了所开发的自动化单元测试工具 AutomatedTest 的特点，以及在企业中实施自动化单元测试所能带来的效益。在第四节详细描述了 AutomatedTest 工具的基于数据驱动的自动化单元测试的方法，并对现有的一些技术作了概括。

第三章　自动化单元测试工具 AutomatedTest 的功能设计

我们设计的自动化单元测试工具 AutomatedTest 是采用了 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单元测试，适用于所有用 C++编码的程序的测试。由于测试是一个需要反复进行的过程，常常需要数十次甚至数百次地重复，而这些非常枯燥而重复的测试方式耗费了大量的时间、人力，并且开发人员一般也不愿意自己进行单元测试此种枯燥的工作，所以开发此自动化单元测试工具来解决此问题，以此来提高工作效率，减轻测试人员的负担，使测试人员的精力更多地投入到高风险区域的识别上。

3.1AutomatedTest 工具的测试原理

AutomatedTest 工具的测试原理：首先选择被测程序，通过对被测试程序的扫描，将测试数据存入 XML 文档和 Excel 工作表中，再根据工作表中的测试数据来构造被测试程序的桩函数和驱动函数，运行测试脚本，产生出测试结果报告。

3.2AutomatedTest 工具的模型设计

3.2.1 需求描述

ATC 系统的软件都是采用 C＋＋或 Microsoft Visual C++6.0 面向对象编程语言或开发工具进行编码的。因此对于测试工具也要求用相同的开发工具，AutomatedTest 使用的就是 Microsoft Visual C++6.0 开发工具。

单元测试是将一个完整的软件划分成若干个独立的测试单元，针对这些测试单元进行隔离测试。自动化单元测试实际上是弥补手工单元测试的不足，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。自动化单元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的单元测试。所以自动化单元测试工具实际上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。AutomatedTest 工具中也是通过此种办法来实现单元测试的自动化的。

单元测试的目的是要在软件开发早期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且要高效。AutomatedTest 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。

对软件进行自动化单元测试，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 AutomatedTest 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，还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，AutomatedTest 工具能够自动精确产生测试数据，并节省了测试时间，使测试人员可以集中精力解决高风险区域的问题。

由此可见，在设计和开发 AutomatedTest 工具时，必须具备的功能主要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。如图 3.2.1.1 所示。整个系统由四个模块组成：被测程序信息提取模块、测试数据存储(或测试数据构造)模块、测试脚本产生模块、测试结果生成模块。

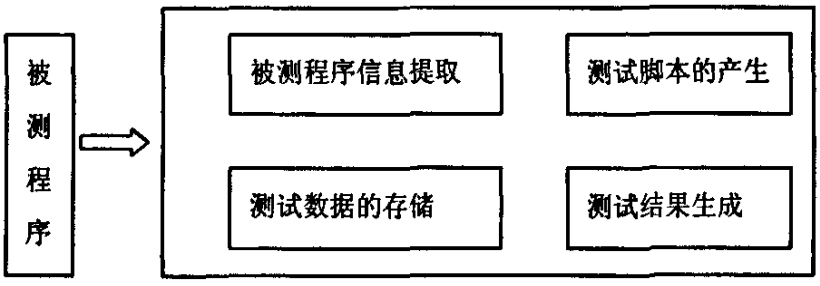


图 3.2.1.1AutomatedTest 功能设计模型图

图 3.2.1.2 是被测程序信息提取模块和测试数据构造模块的执行流程图：

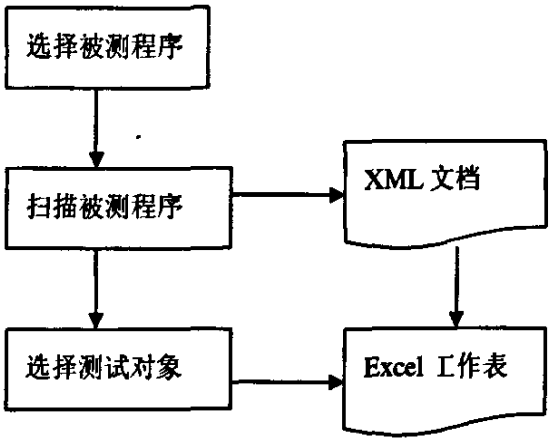


图 3.2.1.2 程序扫描及数据存储流程图

3.2.2 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。

3.2.3 测试数据构造模块

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 AutomatedTest 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。

实现此功能模块的功能，主要是要解决在 XML 文档和 Excel 工作表在 VC++中怎样被操作的问题。

3.2.4 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。从目前的脚本技术来看主要有线性脚本、结构化脚本、共享脚本、数据驱动脚本和关键字驱动脚本。这些技术是相辅相成的，每种脚本在支持完成测试用例的时间和开销上都有自身的特点。AutomatedTest 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 AutomatedTest 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[39]。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。

3.2.5 测试结果生成模块

执行测试脚本时，其测试流程如图 3.2.3.1 所示：

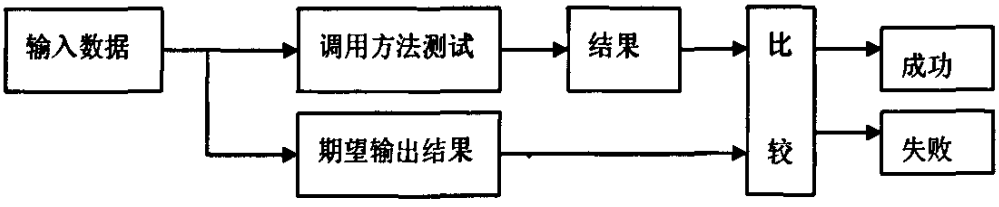


图 3.2.3.1 自动化测试流程

由此可见，此模块的功能较简单，实现也比较容易。只需要将测试结果与期望输出结果作一对比，而后将结果用 Excel 工作表的方式显示出来即可。

3.3AutomatedTest 工具的界面设计

AutomatedTest 工具的界面设计较简单，如图 3.3.1 所示。

图 3.3.1AutomatedTest 工具的主界面

下面对其作一简单介绍：

测试脚本文件夹：此文本输入框中主要用于输入测试脚本存放的文件夹名。

测试结果报表文件夹：此文本输入框中主要用于输入各种数据文档存放的地点。在此框中指定了路径后，所有的 XML 文档和 Excel 电子表格文档都集中存放在此目录下，便于管理和查询。

Start 按钮：是开始执行程序的标志，单击此按钮后，即要求选择被测试程序，然后开始被测程序的扫描和测试数据的存储。

CreateScript 按钮：被测程序扫描完成，测试数据也存储完毕后，则可按此按钮生成测试脚本。脚本执行完成后，自动用 Excel 工作表显示出测试结果。

Exit 按钮：在任何时候停止程序执行，都可按此按钮。

3.4 本章小结

本章描述了 AutomatedTest 工具的测试原理，并根据此原理对 AutomatedTest 工具的功能模块进行了设计。介绍了每个功能模块的功能，以及其中的一些关键要解决的问题。

第四章 AutomatedTest 工具的功能实现

4.1 AutomatedTest 工具的实现环境

硬件配置：CPU：X86

RAM：128MB

硬盘：10G

操作系统：Windows 2000

开发环境：AutomatedTest 工具的开发是采用 Micorsoft Visual C++6.0，语法分析程序采用 ANTLR 开源代码，XML 利用微软的 MSXML 解析器，Excel 是 Office 2003，开发平台是 Windows 2000。

4.2 开发工具的介绍

Windows 是 Microsoft 的产品，在早期阶段，开发工具只有 Microsoft C 和 SDK（Software Developer Kit 软件开发工具包）可供使用。利用 SDK 进行 Windows 程序的设计开发非常繁琐、复杂，代码可重用性差，工作量大，即便一个简单的窗口也需要几百行程序，令开发人员望而生畏。

随着 Windows 的逐渐普及，各大软件公司纷纷推出自己的 Windows 软件开发工具。国内用户比较熟悉的有 Borland C++2.0 以上版本以及用于数据库开发的 Foxpro 等等。其中 Borland C++支持面向对象的开发，在我国具有广大的用户群。

可视化技术和 CASE 技术研究的深入为我们带来了支持可视化编程特性的第三代开发工具，这一代开发工具有：Visual Basic、Visual C++、Borland C++、Builder、Delphi 和用于数据库开发的 PowreBuilder、Visual Foxpro 等等。其中，Visual C++是美国 Microsoft 公司推出的 4GL 软件开发工具，目前已成为国内应用最广泛的高级程序设计语言之一。同其他软件开发工具相比，Visual C++具

有以下优点：

1、面向对象、可视化开发：提供了面向对象的应用程序框架 MFC（Microsoft Foundation Class：微软基础类库），大大简化了程序员的编程工作，提高了模块的可重用性。Visual C++还提供了基于 CSAE 技术的可视化软件自动生成和维护工具 AppWiard、ClassWizard、Visual Studio、WizarBar 等，帮助用户直观的、可视地设计程序的用户界面，可以方便的编写和管理各种类，维护程序源代码，从而提高了开发效率。用户可以简单而容易地使用它进行编程。

2、众多的开发商支持以及业已成为工业标准的 MFC 类库：MFC 类库已经成为事实上的工业标准类库，得到了众多开发商和软件开发工具的支持；另外，由于众多的开发商都采用 Visual C++进行软件开发，这样用 Visual C++开发的程序就与别的应用软件有许多相似之处，易于学习和使用。

3、Visual C++的动态链接库(DLL)的功能，一般情况下，使用 DLL 是为了节省空间，因为 DLL 是在运行期间链接到应用程序，而不是在编译期间。创建一个.exe 文件时，会有许多库例程链接到代码中并放入.exe 文件中。但是 DLL 中的例程则仅仅在实际运行时才链接到.exe 文件，按照这种方法，你可以把几个程序共用的代码放到一个 DLL 中，这样可以节省大量的空间。在开发 AutomatedTest 工具中，就用到了 MS 的一些公用 DLL，如对 XML 文档进行操作的 xml4c 等。

4、Visual C++中的 COM 和 OLE。COM(Component Object Model，组件对象模型)是一种“对象”模型，OLE(Object Linking and Embedding，对象链接和嵌入)和 ActiveX 都建立在它的基础之上。利用 COM，我们能够把对象的函数和数据公布给其他程序供它们使用。OLE 包容器程序可以从 OLE 服务器那里接收 OLE 项。例如，可以把 Microsoft Excel、Microsoft Word 或者其他 OLE 项放到包容器中。

以上这些特点和技术在 AutomatedTest 工具的开发过程中都得到了很好的应用。AutomatedTest 工具的开发采用 Micosoft Visual C++6.0 编程工具。

4.3 被测程序信息的提取功能实现

被测程序信息的提取采用语法分析的开源代码 Antlr[36]。

4.3.1 语法分析

语法分析的依据是语言的规则，即描述程序结构的规则。通过语法分析确定整个输入串是否构成一个语法正确的程序，即对源程序进行语法检查。语法检查的任务就是检查源程序的语法结构是否正确。语法分析程序大致可以分为自顶向下和自底向上两大类。自顶向下分析技术的基本思想是：从识别符号出发，由它推导出与输入符号串相同的终结符号串；而自底向上分析技术与自顶向下刚好相反，从输入符号串出发，试图把它规约成识别符号。

对一种语言程序做语法分析，我们首先必须获得这种语言的语法描述，常常使用 BNF(Backus-Normal Form)范式来描述语言的文法。其实 BNF 很简单::=表示定义，|表示或，尖括号(<>)括起来的是非终结符。所谓非终结符就是语言中某些抽象的概念，终结符就是可以直接出现在语言中的符号。如下面表示了对 C++中标识符的定义：

<标识符> ::= <字母> | <标识符> <字母数字串>

<字母数字串> ::= <字母>|<十进制数字>|<字母数字串><字母>|<字母数字串><十进制数字>

<字母> ::= _ | <大写字母> | <小写字母>

<小写字母> ::= a|b|c|d|e|f|g|h|i|j ……

<大写字母> ::= A|B|C|D|E|F|G|H|I|J ……

4.3.2 Antlr 介绍

ANTLR(Another Tool for Language Recognition)，即语言识别的另一工具，

是一种基于 LL(k)文法的语法分析程序生成工具。其前身是 PCCTS，Purdue Compiler Construction Tool Set，普渡大学编译器构建工具集，由旧金山大学的 Terence Parr 领导开发的。它可以接受语言的文法描述，并能产生识别这些语言的程序。而且我们可以在文法描述中插入特定的语义动作，告诉 ANTLR 怎样去创建抽象语法树(AST)和怎样产生输出。

现在ANTLR越来越流行(有评论说ANTLR的出现是一个里程碑)，不仅因为它功能更强、容易扩展、开源，而且ANTLR生成的代码和使用递归下降方法(手工生成分析器的主要方法)生成的代码很相似，易于阅读理解。最新版本的ANTLR可以去ANTLR的官方网站(http：//www．ANTLR．org)下载。

文法就是语言识别的规则，它是ANTLR生成程序的依据。文法文件是ANTLR的核心，是程序员和ANTLR进行交流的接口。文法文件的编写基本是面向被解决的问题的。程序员只需要集中精力思考解决问题的逻辑，而不是羁绊于某种程序设计语言的实现细节，因此降低了出现错误的可能性。下面对ANTLR的文法文件作一简单介绍：文法文件一般包括header块、options块、文法分析器类（parser）及规则定义、词法分扫描器类（lexer）及token定义。其中最为重要的是规则和token的定义。规则的定义形式和编译理论中的扩展巴科斯范式（EBNF）极为相似，包括规则名、规则体、一个用作结束标志的分号和异常处理部分（可省略）。例如如下的规则就描述了C语言中的赋值语句的语法：

```
assignment_stat:
        id '=' expr ';'
        ;
```

其意义是：一条赋值语句是由一个 id、一个等号、一个表达式和一个分号顺序组成的。

Token 的定义方法与规则类似。例如如下的 token 定义就表示一个十进制的整数：

```
NUM:
        ('1'..'9')('0'..'9')*
        ;
```

其意义是：数字（NUM）的第一字符是‘1’到‘9’中的一个字符，后面是0个或多个‘0’到‘9’之间的字符。

需要注意的一点是：规则的名字必须是小写字母开始，而token的名字则必须是大写字母开始。

ANTLR有很多选项，可以通过在文法文件中的options块中进行设置，其中包括ANTLR最终生成的语言。ANTLR是使用Java开发的，其默认生成的语言是JAVA。如果要生成C++描述的分析器程序，就要如下设定：

```
options
{
language="Cpp";
// Other options
}
```

4.3.3 VC++中的实现

ANTLR在AutomatedTest中的使用，采用了文献[38]中介绍的方法。首先是在AutomatedTest的工程文件中加入ANTLR的源码文件，即点击“Project”—>“Add To Project”—>“Files”，将文法文件test.g添加到工程中。再对此文法文件选择“Settings...”对话框中的“Custom Build”标签页，在“Commands”里填入调用ANTLR编译文法文件命令：java –cp E:\AutomatedTest\antlr-2.7.4\antlr.jar antlr.Tool –o “$(wkspDir)” $(InputName).g。在“Outputs”里面填入ANTLR编译文法文件后要生成的所有文件的名字，如下：ExprLexer.cpp、ExprLexer.hpp 、ExprParser.cpp、ExprParser.hpp、ExprParserTokenTypes.hpp、ExprParserTokenTypes.txt。设置完成后，就可以编译该文法文件了，并生成分析器的源代码，再次将所有生成的文件加入到AutomatedTest工程文件中即可。

4.4 测试数据的存储功能实现

在扫描被测程序时，把获得的被测试程序的信息，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。

4.4.1 用 XML 存储数据

XML 全称是“可扩展标识语言”(Extensible Markup Language)，是 W3C[35] 的一种建议。之所以称之为可扩展，是因为它不像 HTML 那样只有固定的形式。XML 并不是一个独立的，预定义的标识语言。它是一种元语言。它是用来描述其他语言的语言。它允许你自己设计你的标识。C 和 C++和 Fortran, Pascal, Basic, 或是 Java 一样都是编程语言，是用来制定运算和操作的。而 XML 是说明性的语言，是用来表现信息的。它使得信息能被程序正常地存储传输和处理。就它本身而言，并不能产生什么操作。操作要由应用程序来实现。之所以要使用 XML 的原因就是因为我们可以用 XML 设计自己的文件类型。使信息的内容更加丰富，更加方便使用。从组成结构来看，一个 XML 文档包含一个或是多个元素，由一个起始标志符和停止标志符标记其界限，每一个元素有其起始标志符和停止标志符中所指的名称来标识，一个元素可能具有一个值，元素值置于起始标志符和停止标志符之间。

在 AutomatedTest 中，将各种数据信息存储到 XML 文档中是采用了一种较简单的方法，即：被测试程序在编码过程中就由开发人员按照事先约定的格式和内容标注注释，也可以说是以注释形式写出测试用例，这个工作也可以在测试时由测试人员来添加，然后在测试时经过程序扫描来生成测试数据存入 XML 文档中或是 MS Excel 工作表中。此种方法对于开发人员来说，很容易在编码方法时用此种注释形式设定测试用例，而且这种方法能够有效实现后来的测试。另外，使用这种方法，测试人员不必担心何时编写测试数据，从而将减轻测试人员的工作量，可以把更多的精力投入到高风险区域的识别上。在编码实现中，具体使用如下，比如，如果有下列程序代码：

```
double SimpleMath::SimpleCalc(int x,int y,char operation)
{
    double Result;
    switch(operation)
    {
    case '+':
        Result =Add(x,y);
        break;
    case '-':
        Result =Subtract(x,y);
        break;
    case '*':
        Result =Multiply(x,y);
        break;
    case '/':
        Result =Divide(x,y);
        break;
    }
    return Result;
}
```

则在开发人员编写代码时，或者其改动了某个地方，则会加上注释形式的测试用例，如现要对 SimpleCalc 方法进行测试，则在上面程序段前加上以下的注释形式：

```
///<summmary>
///Conduct a simple math operation
///<summary>
///<param name="x">assign an integer,e.g.,245</param>
///<param name="y">assign an integer,e.g.,45</param>
///<param name="operation">assign an char,e.g.,-</param>
```

///<returns>Returns a double result,e.g.,200</returns>

存储在 XML 文档中的内容如下：

```
<member name="SimpleMath.SimpleCalc(Int,Int,char)">
    <summary>
    Conduct a simple math operation
    </summary>
    <param name="x">assign an integer, e.g., 245</param>
    <param name="y">assign an integer, e.g., 45</param>
    <param name="operation">assign an char, e.g., -</param>
    <returns>Returns a double result, e.g., 200</returns>
</member>
```

AutomatedTest 中对 XML 文档的解析，是利用微软的 MSXML 解析器解析 XML 文本，XML DOM（文档对象模型）对象提供了一个标准的方法来操作存储在 XML 文档中的信息，这就是 DOM 应用编程接口（API）函数。它是应用程序和 XML 文档之间的桥梁，DOM 包含两个关键的抽象概念：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。树状层次包括了所有节点，节点本身也可以包含其他的节点。这样的好处是可以通过这个层次结构来找到并修改某一特定节点的信息。MSXML 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，该信息容器被称为节点（NODES）。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知道 XML 的语义。一个文档被解析后，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。在编程时最重要的编程对象是 DOMDocument。 DOMDocument 对象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。

在对 XML 文档进行操作前，要先包含头文件和库：

```
#include <msxml2.h>
#import <msxml4.dll>
```

用 CoInitialize(NULL)来初始化 COM 接口。

初始化 xmlFile 对象：

```
if(FAILED(xmlFile.CreateInstance("Msxml2.DOMDocument.4.0"))) ...
```

然后就可以加载 xml 文件了：

```
_variant_t varXml(L"C:\\test.xml"); //L for unicode
    VARIANT_BOOL varOut;
    xmlFile->load(varXml, &varOut);
```

取得 root element：

```
xmlFile->get_documentElement(&xmlRoot))
```

取得第一级 element：

```
IXMLDOMNodeList* xmlChildNodes = NULL;
    xmlRoot->get_childNodes(&xmlChildNodes);
```

遍历所有第一级 element:

```
IXMLDOMNode* currentNode = NULL;
    while(!FAILED(xmlChildNodes->nextNode(&currentNode)) &&
    currentNode != NULL)
    {
    //do something
    }
```

取得当前 element 的名称：

```
BSTR nodeName;
    currentNode->get_nodeName(&nodeName);
```

取得当前 element 的一个 attribute 的值：

```
IXMLDOMNamedNodeMap* attributes = NULL;
    IXMLDOMNode* attributeName = NULL;
    _bstr_t bstrAttributeName = "class";
    BSTR nameVal;
    currentNode->get_attributes(&attributes);
    attributes->getNamedItem(bstrAttributeName, &attributeName);
    attributeName->get_text(&nameVal);
```

最后用 Release()释放所有的接口。

4.4.2 使用 Excel 工作表存储数据

XML 文档的作用只是作为存储数据使用。在 AutomatedTest 的设计中，并不是每次测试都要将整个被测程序进行全部测试，而是设计了一个选择框来对要测试的内容进行选择，只有要测试的类才写其测试用例，否则就不需要写测试用例。所以在 AutomatedTest 中采用了 Excel 工作表来存储测试用例，这样做的好处是：将测试数据在进行测试输入之前，能够方便地进行修改。在 AutomatedTest 工具中采用了 Excel 工作表来进行存储测试数据，以便能够随时修改测试数据的一些值，以此来很方便地就能做到对同一方法的多次测试。

在 VC++6 中要能够实现以编程的方式对 Excel 工作表进行操作，则必须先在工程中添加上该组件对象。具体操作如下：

在 view 菜单里面选 classwizard，然后选 Automation 面板，再按 Add Class，选"From a type library." ，然后再选需要的 object library。如图 4.4.2.1 所示。

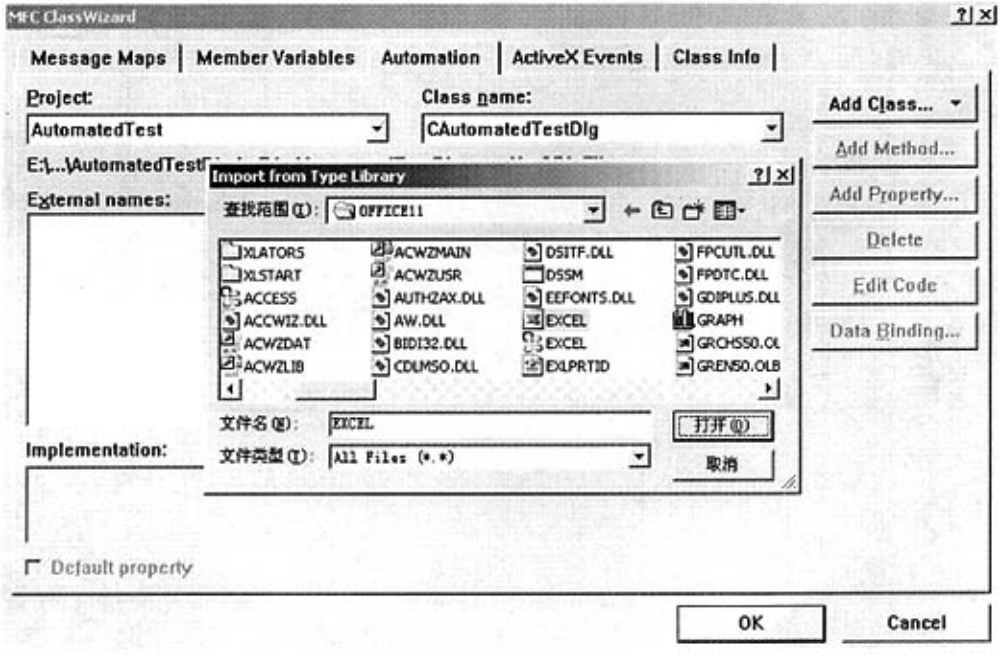


图 4.4.2.1 在工程中添加 Excel 的组件对象

在 Office 2003 中 Excel 的组件对象文件是“C:\program Files\Microsoft Office\Office11\Excel.exe”。不同的 Office 版本其文件是不相同的，如 Excel 97 中要用文件 Excel8.olb，Excel 2000 中的文件是 Excel9.olb，而 Microsoft Excel 2002 和 Microsoft Office Excel 2003 中都用文件 Excel.exe，只是 2002 中文件路径是在 Office10 下。

在图 4.4.2.1 中按“打开”后，则出现如图 4.4.2.2 所示的确认类对话框，在此对话框中选择要对其操作的 Excel 的类。添加完后，会在工程文件目录下自动生成文件 excel.h 和 excel.cpp，编程时包含头文件即可。

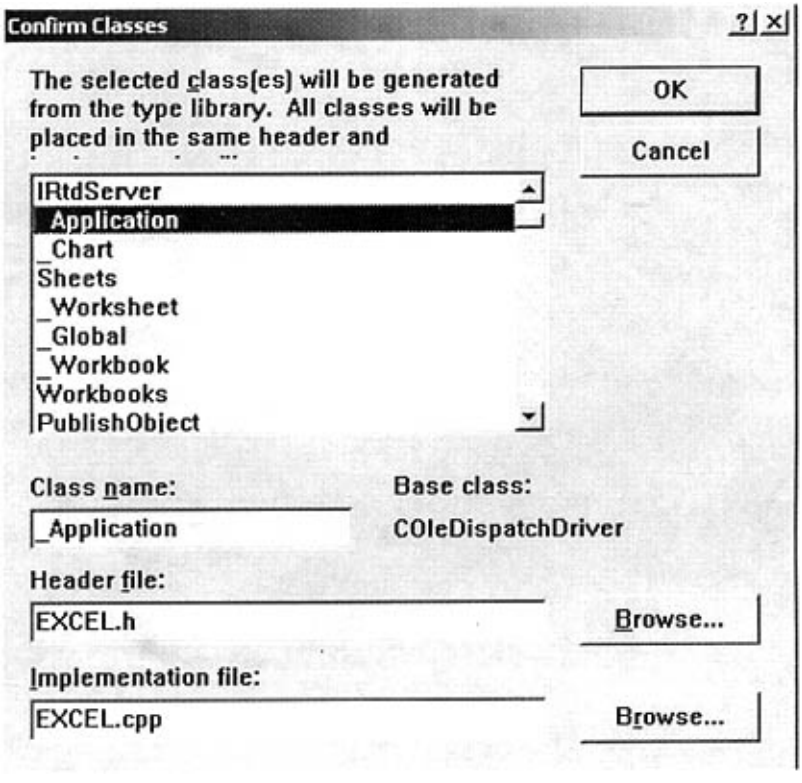


图 4.4.2.2 选择 Excel 的类

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

在 VC++中，_Application 类，它是对象类，即要在 VC++中对 Excel 这个应用程序对象进行操作，则必须先建立这个对象，才能对其所包含的属性、方法和事件等进行操作，建立 Excel 对象：

```
_Application *ExcelApp = new _Application;
```

然后通过如下编码产生一个工作簿：

```
Workbooks  ExcelBook=ExcelApp->GetWorkbooks();
```

不能直接使用工作簿或应用程序进行数据修改，需要为 Workbooks 对象增加电子数据表来存储信息。下面的代码为工作簿增加一个工作表，工作表是 Excel 应用程序存储信息的位置：

```
Sheets    ExcelSheet=ExcelBook.Add(vtOptional);
```

这里产生的对象是隐藏的，为了使得对象可见，设置应用程序的 Visible 属性值为 TRUE：

```
ExcelApp->SetVisible(TRUE);
```

通常打开一个应用程序之后，需要用键盘或鼠标来完成一些任务。然而，我们在 AutomatedTest 中是不用人工干预，让 Excel 自己独立完成一些任务，因此，需要了解如何操纵 Excel，并基于 VC++，编码完成对 Excel 应用程序方法的操作。下面简要介绍工作表数据存储的一些方法：

GetActiveCell()，该方法返回当前焦点单元格的 Range 对象。

GetActiveSheet()，该方法返回当前工作表的 Range 对象。

GetActiveWorkbook()，该方法返回当前工作簿的 Range 对象。

在 Excel 应用程序中打开一个工作簿后，可以用如下方法对其进行操作：Activate()方法用于激活工作簿，Add()方法用于产生新的工作簿，Close()方法用于关闭工作簿，Open()方法用于打开现存工作簿。如下面代码即用于打开一现存工作簿：

```
CString workbookPath = "E:\\AutomatedTest\\excel.xls";
ExcelBook.Open(
```

```
        workbookPath,                    //FileName
        COleVariant((long)0,      VT_I4),      //UpDateLinks
        COleVariant((long)FALSE,VT_BOOL),      //ReadOnly
        COleVariant((long)1,      VT_I4),      //Format
        COleVariant(""),                 //Password
        COleVariant(""),                 //WriteResPassword
        COleVariant((long)FALSE,VT_BOOL),
//IgnorReadOnlyRecommended
        COleVariant((long)2,      VT_I4),          //Origin ,2:xlWindows
        COleVariant(),                   //Delimiter
        COleVariant((long)FALSE,VT_BOOL),      //Editable
        COleVariant((long)FALSE,VT_BOOL),      //Notify
        COleVariant(),                   //Converter
        COleVariant((long)FALSE,VT_BOOL)       //AddToMru
    );
```

工作表对象 Worksheet 是 Worksheets 集合的成员，Worksheets 集合代表工作簿的所有表单，通过编程可以改变 Worksheet 对象的值、颜色、行、列属性。工作表方法 CheckSpelling()用于检查工作表的文本拼写，Copy()方法用于复制指定的工作表，Delete()方法用于从 Workbook 对象中删除工作表，Move()方法用于移动 Worksheet 对象指向的工作表。

正如我们所知道的，工作表是数据存放的区域。Excel 中的 Range 对象是工作表的一部分。包括通过键盘、鼠标或 VC++代码行指定的一个或多个单元格。

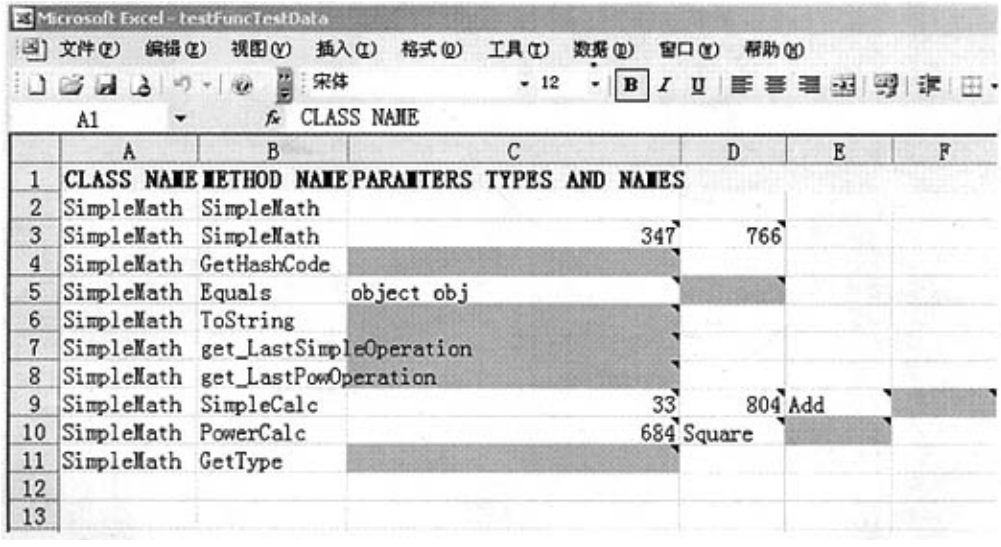
以下代码行实现了在指定单元格“A2”中存储数据“CLASS_NAME”的功能：

```
Range
myrange=workSheet.GetRange(COleVariant("A2"),COleVariant("A2"));
myrange.SetValue(COleVariant("CLASS_NAME"));
```

以下代码是对 A2 单元格中显示的数据格式进行的设置：

```
myrange.BorderAround(COleVariant((short)1),(long)2,(long)1,vtOptional);
myrange.SetHorizontalAlignment(COleVariant((short)3));
myrange.SetVerticalAlignment(COleVariant((short)2));
myrange.SetColumnWidth(COleVariant((short)15));
```

通过对以上应用程序对象 Application，工作簿对象 Workbook，工作表对象 Worksheet，以及区域对象 Range 的属性、方法等的操纵，用 VC++即可以实现将数据存入 Excel 工作表的功能。产生的工作表如图 4.4.2.3 所示。



	A	B	C	D	E	F
1	CLASS NAME	METHOD NAME	PARAMTERS TYPES AND NAMES			
2	SimpleMath	SimpleMath				
3	SimpleMath	SimpleMath	347	766		
4	SimpleMath	GetHashCode				
5	SimpleMath	Equals	object obj			
6	SimpleMath	ToString				
7	SimpleMath	get_LastSimpleOperation				
8	SimpleMath	get_LastPowOperation				
9	SimpleMath	SimpleCalc	33	804	Add	
10	SimpleMath	PowerCalc	684	Square		
11	SimpleMath	GetType				
12						
13						

图 4.4.2.3 用 Excel 工作表存储的数据

此表中列出的数据是所选择来要测试的类的数据。AutomatedTest 工具已经为参数赋予了合适的值，用户可以选择保持或更改原参数值。若选择保持原参数值，则可以直接完成测试；否则，测试脚本产生前，需要暂停测试执行。这时，用户可以编辑测试数据存储，数据编辑确认后，继续测试执行，编写测试脚本，提交测试结果。

如果要使用在编码时写入被测程序测试用例中的值，则还需要做的操作是将 XML 文档中存储的相应数据项值取出覆盖 Excel 表中相同数据项值即可。这个功能通过一个 for 循环即可实现。

4.5 测试脚本生成功能实现

软件测试的主要任务是编写测试脚本，用于测试被测程序中所有的类、方法和属性。

AutomatedTest 中是利用存储在 Excel 表中的数据信息来自动生成变量控制函数、桩函数和驱动函数。包含在文件 Driver.cpp 中，根据工作表中存储的方法和全局变量的类型来生成相应的程序，然后与结果生成部分脚本程序一起编译执行。

Driver.cpp 包含以下几部分：通过 extern 引用全局变量和桩函数；全局变量赋值和取值，通过方法 SetGlobalVar 和 GetGlobalVar 完成；桩函数(多个)；驱动函数(多个)。

为了简化自动生成的函数，使其易于理解，AutomatedTest 定义了很多简化程序结构的宏，如 FUNC_COPYTOVAR(a)就是一主要的宏，表示将 Excel 工作表中的单元格数据赋值给变量，赋值操作用 memcpy 完成，这样使得这些宏能够对所有类型通用，在变量和指针的赋值中都能编译通过。

通过很多这样类似的宏，驱动函数和桩函数的自动生成已很容易做到。

4.6 测试结果生成功能实现

测试脚本如何验证测试结果，如何描述测试结果，是此功能模块中要解决的核心问题。

4.6.1 验证测试结果

验证测试通过计算机硬、软件接口，执行测试脚本，基于测试结果验证如下几方面的内容：

被测类各成员的执行是否正常；

执行环境是否和指定的硬、软件配置相匹配；

执行过程中是否捕获到例外；

执行过程中是否产生错误信息。

总之，验证测试用于验证代码在预定义条件集合下的执行是否正确。

在 AutomatedTest 工具的开发中，用方法 TestPass()和 TestFail()来描述测试的成功和失败的信息，验证测试是用 try-catch 语句实现的，如下所示：

```
…
Try{
    …
    TestPass(…);
}
Catch(…) {
     TestFail(…);
}
…
```

在 try 语句体中的代码能够正常执行，则调用 TestPass()方法，确保代码执行中没有出现逻辑错误；若 try 语句体中的代码执行失败，则执行 catch 语句体中的代码，报告失效原因。这样 try-catch 语句块中调用了关键的测试行为。

验证测试过程中，catch 语句体能够捕获错误信息，从而能够达到如下的目标：

提供关于产品质量和执行效率的数据；

揭示缺陷；

提高软件测试的有效性。

测试人员的任务是尽可能早地揭示尽可能多的错误，然后，开发人员基于测试报告修正揭示的错误。很明显，测试/修正过程在软件开发过程中的连续性并不能保证软件中没有错误。例如，当测试带整数类型的参数的方法时，不可能测试所有有效整数以及这些整数参数在各种环境下的组合。AutomatedTest 使用边界条件策略，则测试人员就需要做如下的工作：

彻底审查产生的数据存储，深入理解每一条信息；

选择更容易揭示潜藏缺陷的测试用例值，而不是能使程序正确工作的测试用例值，保存到 Excel 工作表的数据存储中；

Excel 工作表的数据存储中既有有效数据，也有无效数据，用于测试被测方法能否返回期望结果或错误信息；

利用期望返回值验证测试结果；

使用其他测试方法运行同样的测试用例；

尽可能寻找和产品质量相关的软件错误。

4.6.2 测试结果描述

描述测试结果，为开发人员提供修正缺陷所需的信息。在 AutomatedTest 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，另外一类测试结果描述真实执行返回值和期望返回值是否相匹配，这些都由测试脚本中的 TestPass()和 TestFail()方法来处理。最终得到的 Excel 报告包括了如下的测试结果：

第一列代表被测方法的名字，其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；

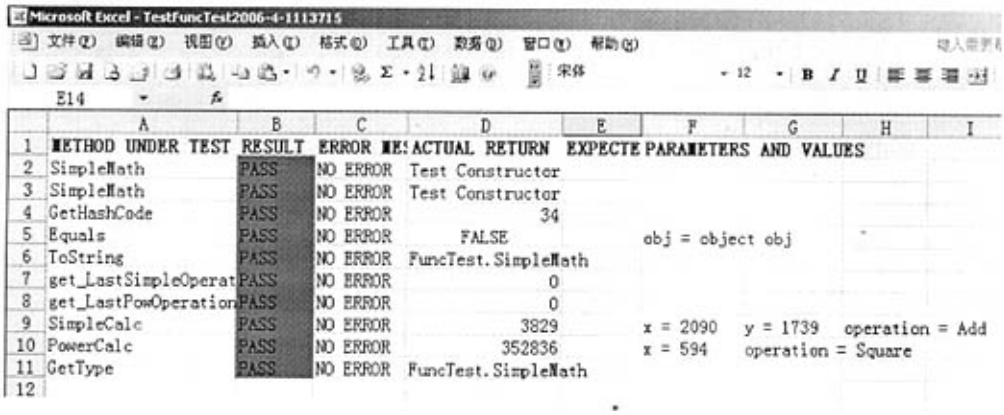
第三列报告错误消息，若测试失败，提示失败的原因，这些消息来自不同的计算资源；

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；

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

具体报告如表 4.6.1 所示：



Microsoft Excel - TestFuncTest2006-4-1113715

	A	B	C	D	E	F	G	H	I
1	METHOD UNDER TEST	RESULT	ERROR ME:	ACTUAL RETURN	EXPECTE	PARAMETERS AND VALUES			
2	SimpleMath	PASS	NO ERROR	Test Constructor					
3	SimpleMath	PASS	NO ERROR	Test Constructor					
4	GetHashCode	PASS	NO ERROR	34					
5	Equals	PASS	NO ERROR	FALSE		obj = object obj			
6	ToString	PASS	NO ERROR	FuncTest.SimpleMath					
7	get_LastSimpleOperat	PASS	NO ERROR	0					
8	get_LastPowOperation	PASS	NO ERROR	0					
9	SimpleCalc	PASS	NO ERROR	3829		x = 2090	y = 1739	operation = Add	
10	PowerCalc	PASS	NO ERROR	352836		x = 594	operation = Square		
11	GetType	PASS	NO ERROR	FuncTest.SimpleMath					
12									

表 4.6.1 测试结果报告

4.7 本章小结

综上可见，所开发的 AutomatedTest 工具的特点主要表现如下：

第一：通过对被测代码进行扫描，获取程序信息来做到测试数据的构造，而不以在编译时引用被测代码中的结构等信息方式实现，从而做到数据的构造在编译层次上独立于被测程序，避免因测试数据的变化而导致相应测试代码的变化和重复编译。

第二：脚本和数据的分离技术，能避免因增加新的测试数据或是测试数据的变化而导致书写新的测试脚本或是修改现有测试脚本，使得测试脚本在开始书写后只对测试数据编辑即可，简化了白盒测试的实现，实现数据驱动测试脚本执行，提高了白盒测试的效率。

本章详细介绍了自动化单元测试工具 AutomatedTest 各个功能模块的实现过程。被测程序信息提取主要是用语法分析的开源代码 ANTLR 来实现；测试数据的存储主要是用 XML 文档和 Excel 工作表来进行，文中给出了详细的解决过程；测试脚本的生成是根据工作表中存储的数据信息来自动生成被测程序的桩函数的驱动函数，文中介绍了用宏来实现这一功能的具体方法；测试结果的生成主要是将测试结果写入 Excel 工作表的过程。

第五章　用 AutomatedTest 实现自动化单元测试

用 AutomatedTest 工具进行自动化测试时主要有 6 个步骤，如图 5.1 所示。根据回归测试的定义，软件开发生存周期中，需要不断测试经常改变的代码，检测并修正缺陷后，回归测试用于确认对缺陷的修改精确，且没有负面影响。因此，图 5.1 中迭代是以回归测试结束的。

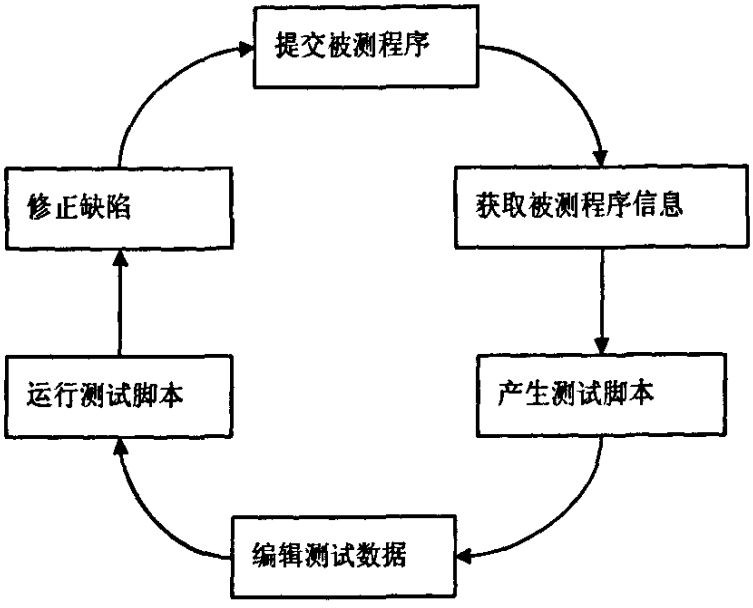


图 5.1 自动软件测试的 6 个步骤

下面以一个完整的自动化单元测试过程来验证和检测 AutomatedTest 工具所具有的功能。

5.1 环境配置

AutomatedTest 工具是用 Microsoft Visual C++6.0 开发的，Windows 2000 操作系统，另外还需要 Microsoft Excel 2003。AutomatedTest 工具的源代码准备好后，用调试配置或发行配置编译源代码。即可生成 AutomatedTest.exe 文件。

5.2 提取被测程序信息

运行 AutomatedTest.exe 文件，其主界面是一对话框，如图 5.2.1 所示：

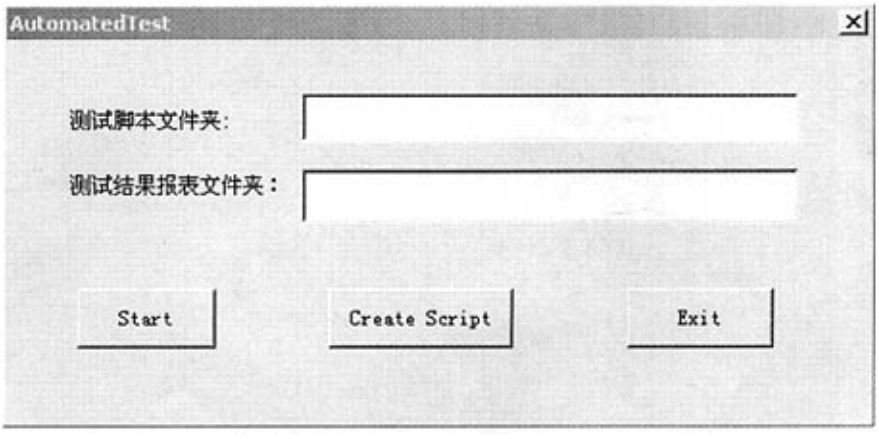


图 5.2.1 AutomatedTest 主界面

在此界面中，先要设定生成的测试脚本所要存放的文件夹，以及存放测试结果报表的文件夹。以便程序执行过程中将所产生的文件按事先的约定存入指定的地点，如事先没有此文件夹，还要先建立它。如图 5.2.2 所示输入文件夹名：

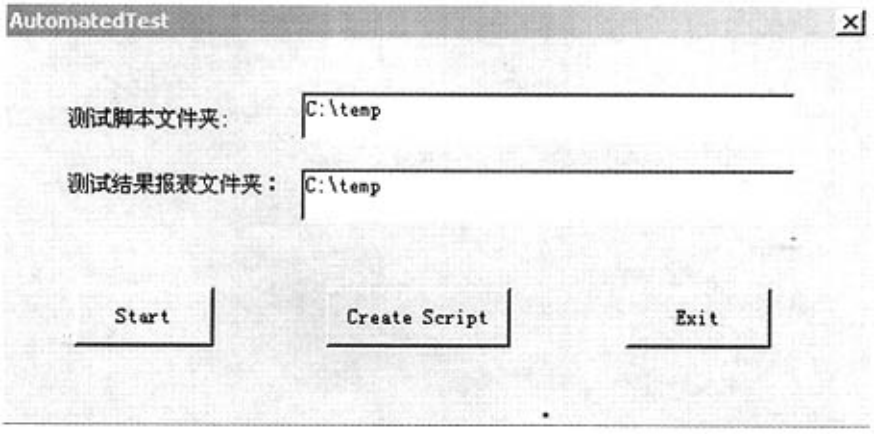


图 5.2.2 指定存储文件的文件夹界面

然后，按 Start 按钮，会出现如图 5.2.3 所示的打开文件对话框，此对话框只能选择.CPP 文件，因为测试工具是只针对 CPP 文件的测试工具。

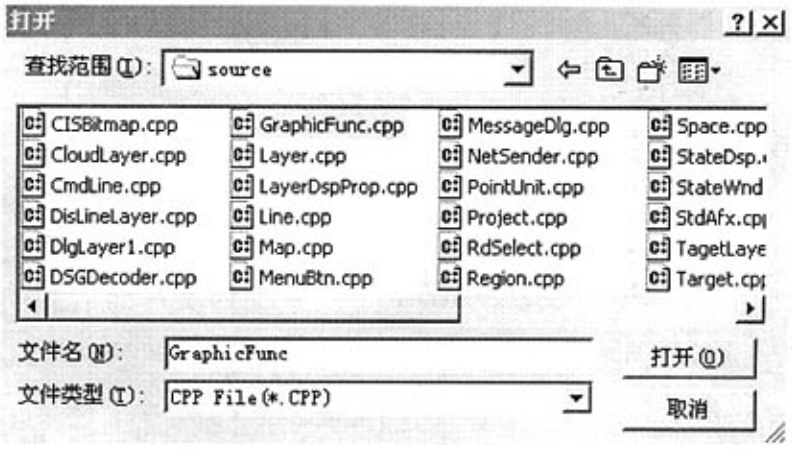


图 5.2.3 打开 CPP 文件对话框

在此打开文件对话框中，选择所要进行测试的程序，并单击“打开”。此时程序将开始对所选择的被测程序进行扫描，将测试数据存入 XML 文档中，并在如图 5.2.4 所示的 SelectedMethods 对话框的左边显示出所扫描到的被测试程

序中的所有类名，在此对话框中可以根据实际需要来选择感兴趣的类进行测试，而不必要全部测试。在对话框左边列表框中选出要测试的类，单击“select”则在右边列表框中列出要进行测试的类名，然后再按“OK”按钮继续下面操作。

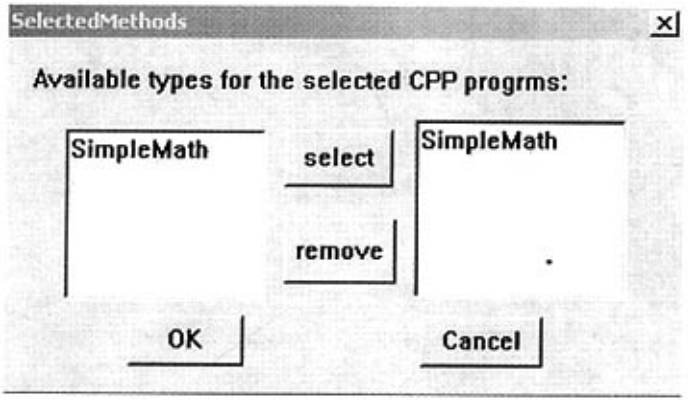
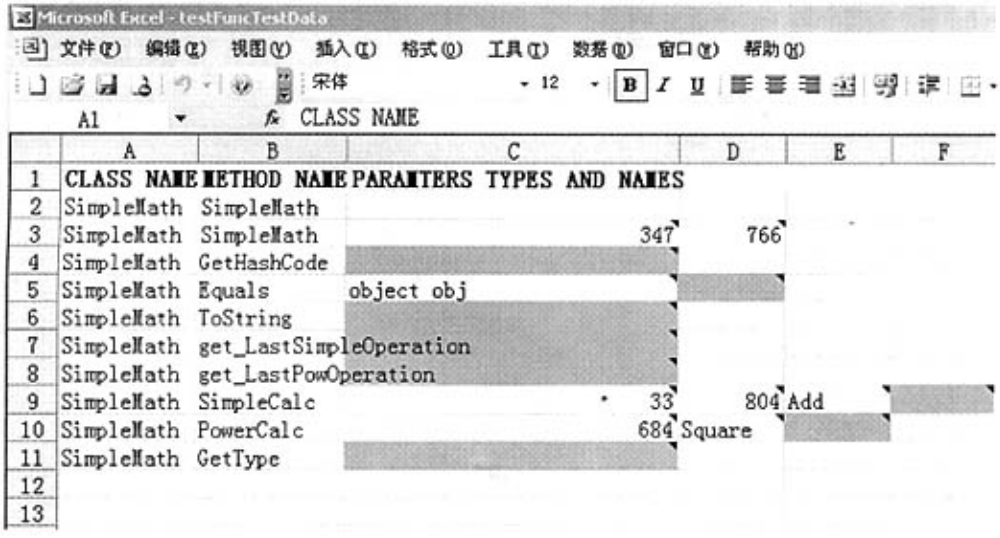


图 5.2.4 选择被测内容对话框

在选择了要测试的类并按“OK”后，程序打开了一个 MS Excel 工作表。AutomatedTest 工具开始收集 SimpleMath 类的信息，信息收集完成后，MS Excel 工作表的内容如图 5.2.5 所示。工作表的文件名是用“test”+被测程序文件名＋“Data”构成，这种命名方法以用于区分结果报表文件名。图 5.2.5 所示的 Excel 工作表包括用于编写和运行测试脚本的许多信息，测试人员可以随时访问这些信息。



	A	B	C	D	E	F
1	CLASS NAME	METHOD NAME	PARAMTERS TYPES AND NAMES			
2	SimpleMath	SimpleMath				
3	SimpleMath	SimpleMath	347	766		
4	SimpleMath	GetHashCode				
5	SimpleMath	Equals	object obj			
6	SimpleMath	ToString				
7	SimpleMath	get_LastSimpleOperation				
8	SimpleMath	get_LastPowOperation				
9	SimpleMath	SimpleCalc	33	804	Add	
10	SimpleMath	PowerCalc	684	Square		
11	SimpleMath	GetType				
12						
13						

图 5.2.5 AutomatedTest 所收集的被测类的信息

到此为止，提取被测程序信息的操作就已做完。其中，XML 文档是存放在与被测程序相同的目录下。文件名与被测程序的文件名相同。

5.3 编辑测试数据

图 5.2.5 的 Excel 工作表的第一列是类名。由于只有一个测试类型类 SimpleMath，因此所有行的第一列都是同一个类名。第二列列出了从 SimpleMath 类中提取出的构造函数、属性和方法。这里可以通过复制、粘贴或删除，确定需要测试哪个方法，以及设置测试方法的次数。类的构造函数名总是列在最前面，若构造函数重载了不止一次，则至少测试一次构造函数。从第三列开始，若被测方法带参数，则列出参数。当参数是数值类型时，AutomatedTest 工具自动赋给该参数一个随机数。当然，可以随时修改这个值。

MS Excel 工作表具有确认属性，能够列出单元格的可能值，如图 5.3.1 所示。SimpleMath 类的 PowerCalc（）方法带枚举参数 Power_ENUM，其有三个

可能的值：Square,Cubic 和 SquareRoot，单击工作表单元格可以激活确认属性。如图 5.3.1 所示，单击单元格后，单元格的右边出现一个箭头，单击箭头，出现的列表枚举了类型的可能值，可以从列表中选择要使用的数学运算符，工具给的默认值是 Square。因此，当参数是枚举类型时，工作表弹出一个下拉列表，列表中包括了所有的值，可以选择使用其中的某些值。使用 MS Excel 工作表的确认属性，测试人员不会错误地输入无效值，特别对于枚举类型包括很多元素时，该属性对编辑数据存储尤其有用。若想用无效参数值测试被测方法，可以把任何无效值输入工作表单元格。

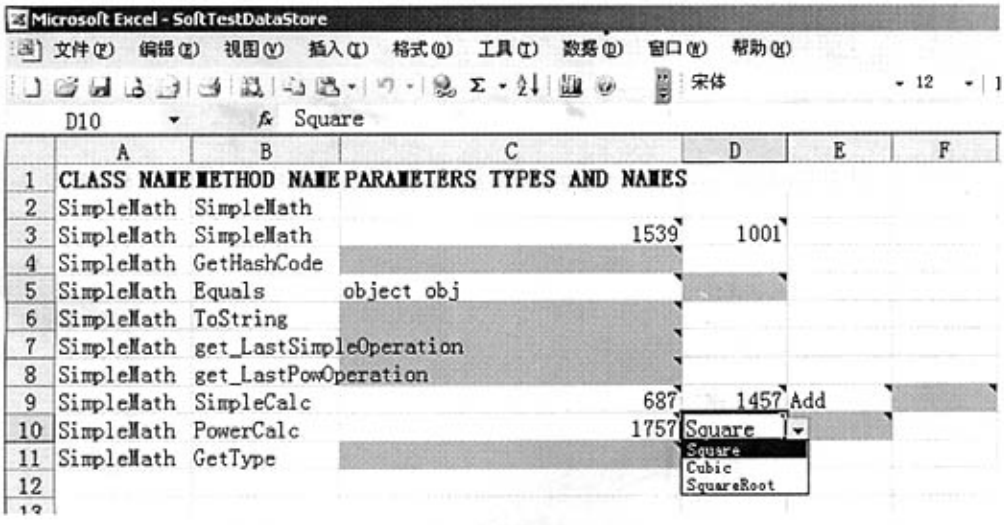


图 5.3.1　一个单元格的多个值

每个参数单元格右上角的红三角用于提示信息。如图 5.3.2 所示鼠标指针停留在参数单元格上时，弹出一个小窗体，显示该单元格的参数类型和参数名字，这就是 MS Excel 工作表的 Comment 属性，该属性有助于将来再次修改数据表单时输入正确的参数类型。图中的注释表示 1457 是整型变量 y 的值。

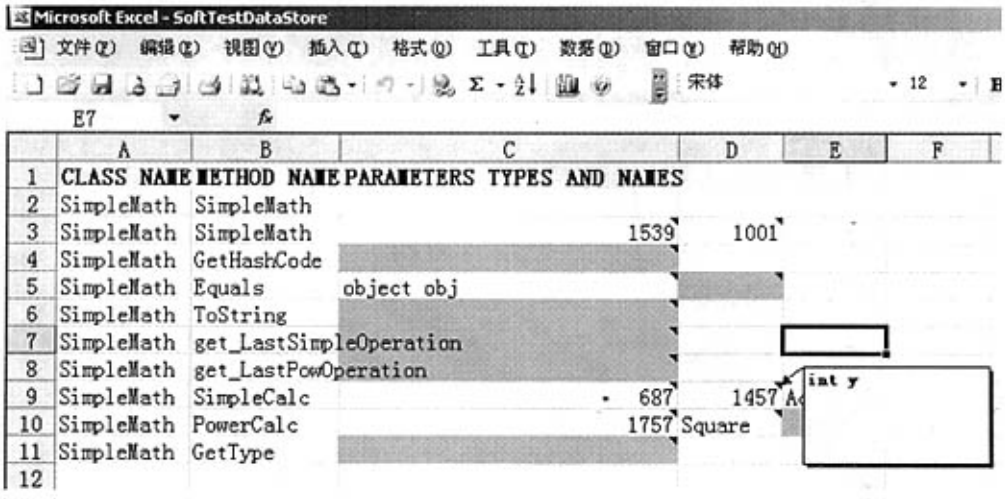


图 5.3.2 参数的注释窗体

当参数通过值传递时，若默认参数值的检错能力不强，则可以改变参数值。若参数类型是数值数据类型，不要在单元格中输入字母，否则，测试工具会产生一条错误消息，然而，若想测试被测程序对无效输入的错误处理能力，可以在单元格中输入无效数据。若参数类型是字符串类型，则可以选择不修改原来的赋值，也可以选择赋予参数合适的文本值。

若要用不同的参数值或测试用例多次测试同一个方法，可以复制这个方法所在的行，粘贴多次，并输入不同的参数值。若不想测试某个方法，可以删掉该方法所在的行。如图 5.3.3 所示。但必须记住的是，被删方法间不能存在任何空行，因为 AutomatedTest 工具会在遇到的第一个空行处停止执行。

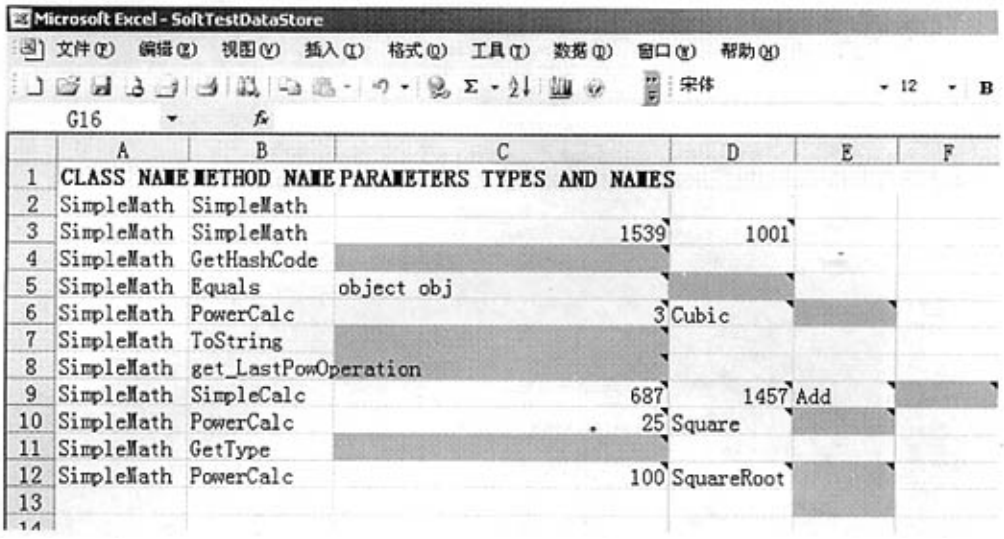


图 5.3.3　参数赋值以及复制或删除一些被测方法后的数据表单

在图 5.3.3 中，复制了 PowerCalc()方法并重新为参数赋了值。删除了一个方法。可见，用此方法很方便地就能对所指定的方法进行重复测试。

另外一条多次测试同一个方法的途径是：以其他文件名保存 Excel 工作表的多个备份，然后，在每个工作表中，为参数赋予不同的值，但不同工作表中的方法所在的行必须相同。

5.4 执行测试用例

在确定了测试数据之后，再回到主界面对话框中按“Create Script”按钮，如图 5.4.1 所示。自动产生桩函数和驱动函数到指定的文件夹目录下。将程序编译运行即可得到测试结果报表。

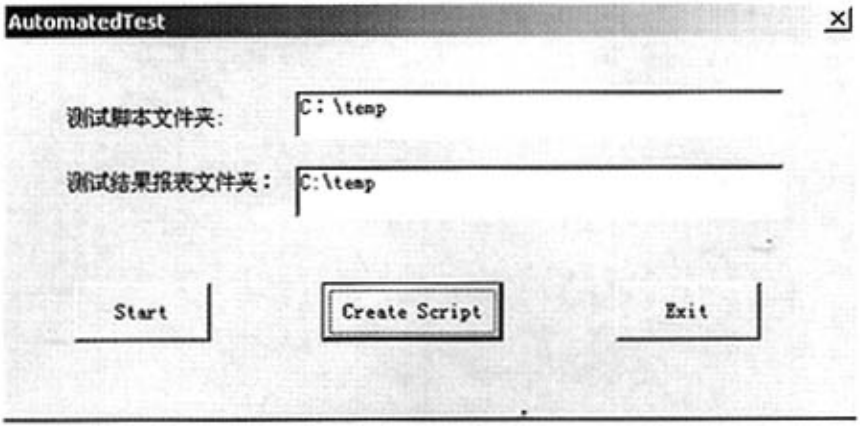
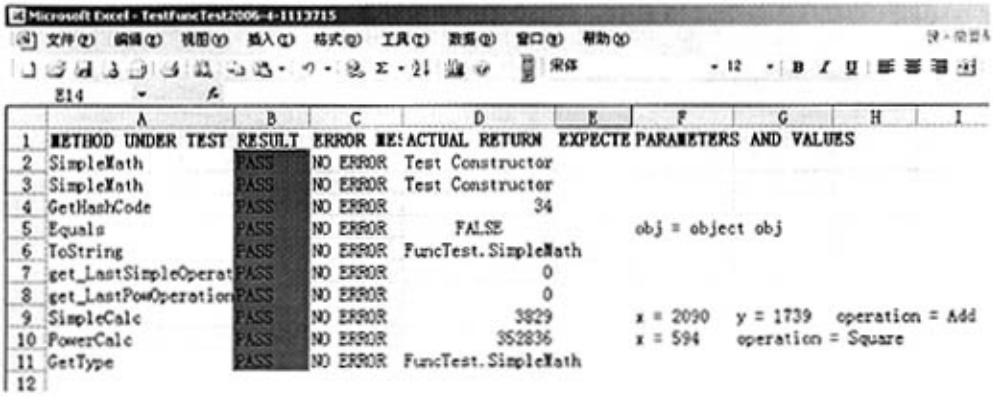


图 5.4.1 主界面对话框

5.5 查看测试结果

测试脚本运行后，会在指定的文件夹下生成结果文件，其文件名的命名方法是“Test”+被测程序文件名＋测试的日期和时间，如图 5.5.1 所示。在此报告中，第一列列出了被测方法的名字；第二列列出了验证测试结果，值为 Pass 或 Fail；若值为 Fail，则第三列报告一条错误消息，指出失效原因；若被测方法有返回值，则第四列列出真实返回值；第五列是期望返回值。



	A	B	C	D	E	F	G	H	I
1	METHOD UNDER TEST	RESULT	ERROR ME	ACTUAL RETURN	EXPECTE	PARAMETERS AND VALUES			
2	SimpleMath	PASS	NO ERROR	Test Constructor					
3	SimpleMath	PASS	NO ERROR	Test Constructor					
4	GetHashCode	PASS	NO ERROR	34					
5	Equals	PASS	NO ERROR	FALSE		obj = object obj			
6	ToString	PASS	NO ERROR	FuncTest.SimpleMath					
7	get_LastSimpleOperat	PASS	NO ERROR	0					
8	get_LastPowOperation	PASS	NO ERROR	0					
9	SimpleCalc	PASS	NO ERROR	3829		x = 2090	y = 1739	operation = Add	
10	PowerCalc	PASS	NO ERROR	352836		x = 594	operation = Square		
11	GetType	PASS	NO ERROR	FuncTest.SimpleMath					
12									

图 5.5.1 Excel 工作表的报告

现在已经完成了测试，同时产生了测试脚本文件，以后可以随时用不同的测试用例重新运行测试。

5.6 本章小结

本章演示了 AutomatedTest 工具的测试功能。通过这些演示，我们可以看出 AutomatedTest 工具作为单元测试工具在测试数据的构造和测试用例的创建、测试脚本的生成等方面相比一般测试工具还是有了比较大的改进，自动化程度得到了一定程度的提高。

第六章 结束语

软件测试是软件工程的重要环节，它直接关系到软件的质量、开发进度和成本。软件测试作为软件工程学科的一个分支，软件测试做得怎样，决定着软件产品质量的好坏。随着软件技术和编码技术的巨大发展，软件设计和编码的效率得以极大的提高。软件测试的工作量相比过去并未减少，相反地，在整个软件的生命周期中，软件测试工作所占的比例不断的提高。自动化测试的出现和发展使得软件测试进入了一个崭新的时期，而自动化测试技术和测试工具是提高自动化测试水平和效率的基本手段。

随着软件工程的迅猛发展，人们期待着自动化测试技术能完全取代传统的手工测试，然而至今为止，还没有任何一种自动测试工具可以实现这一目标。软件测试研究开发人员一直在努力提高自动化测试技术，推动软件工程的发展。软件自动化测试领域的发展与每一位研究人员的努力都是息息相关的。

本文在对软件测试作了充分的研究基础之上，自主开发出了针对 C++程序的自动化单元测试工具 AutomatedTest。使其具有如下功能：自动提取被测程序信息，自动存储测试数据，根据测试数据自动生成桩函数和驱动函数，执行测试脚本后，自动生成测试结果报表。

虽然具有了这些功能，但此工具还有很多地方值得改进：界面过于简单，交互性不好；对内存泄漏或复杂指针操作等问题显得无能为力，功能也不够强大，与当前的主流商用测试工具还有很大的差距，所有这些问题还需要进行不懈的努力和改进。

到目前为止，自动化测试技术还处于对测试人员要求比较高，自动化程度不够的阶段。随着计算机技术、软件技术等的进一步发展，通过人们坚持不懈的努力探索，自动化测试技术必将会得到很大的发展。自动化测试工具其功能将越来越强大，适用面会越来越广。

参考文献

[1]Myer G. The Art of Software Testing. Wiley,1989

[2]郑人杰，殷人昆，陶永雷． 实用软件工程[M]. 北京：清华大学出版社，2000

[3]张海藩编著． 软件工程导论[M]. 北京：清华大学出版社，2003

[4]林宁，孟庆余主编. 软件测试实用指南[M]. 北京：清华大学出版社，2004

[5] 古乐，史九林编著. 软件测试技术概论[M]. 北京：清华大学出版社, 2004.

[6]（美）Paul C. Jorgensen 著，韩柯，杜旭涛译. 软件测试[M]. 北京：机械工业出版社，2003.

[7]（美）Elfriede Dustin, Jeff Rashka, John Paul 著. 自动化软件测试－入门、管理与实现[M]. 北京：清华大学出版社，2003

[8] （美）Elfriede Dustin, Jeff Rashka, John Paul 著.于秀山等译. 软件自动化测试－引入、管理与实施[M]. 北京：清华大学出版社，2003

[9] (美)[M.菲斯特](Mark Fewster)，(美)[D.格雷厄姆](Dorothy Graham)著. 舒智勇等译；软件测试自动化技术与实例详解[M]. 北京：电子工业出版社, 2000.

[10] 飞思科技产品研发中心. 实用软件测试方法与应用[M]. 北京：电子工业出版社，2003

[11] http://www-306.ibm.com/software/awdtools/suite/index.html.

[12] http://www.mercury.com/us/products/.

[13] http://www.compuware.com/products/devpartner/studio.htm.

[14] http://www.parasoft.com/jsp/products/home.jsp?product=Insure.

[15] http://www.obsoft.com/Product/CPlusPlus.html.

[16] http://www.segue.com/.

[17] http://www.soft.com/Products/index.html.

[18] 程萃，OOT 在 ATC 中的应用. 四川大学硕士学位论文，2004

[19]郭小华，ATC 系统测试的研究与实现. 四川大学硕士学位论文，2005.

[20]（美）莫斯里(Mosley,D.J.)等著；邓波，黄丽娟，曹青春等译. 软件测试自动化[M]. 北京：机械工业出版社，2003

[21]蔡军红.软件质量与测试（三）.软件世界，2002，（8）：109~112.

[22]R. Ferguson, B. Korel. The Chaining Approach for Software Test Data Generation. ACM Trans. Softw. Eng. Method，1996,5（1）：63~68

[23]荚伟，高仲仪. 基于遗传算法的软件结构测试数据生成技术研究. 北京航空航天大学学报，1997,23(1)：36~40

[24]A. Hajnal, I. Forgacs. An Applicable Test Data Generation Algorithm for Domain Errors. ACM SIGSOFT Software Engineering Notes. 1998, 23(2): 63~72

[25]N. Gupta, A. P. Mathur, M. L. Sffa. Automated Test Data Generation Using An Iterative Relaxation Method. ACM SIGSOFT Software Engineering Notes. 1998, 23(6): 231~244

[26]B. Korel. Automated Test Data Generation for Programs with Procedures. ACM SIGSOFT Software Engineering Notes. 1996, 21(3): 209~215

[27] S. Rapps, E. J. Weyuker. Data Flow Analysis Techniques for Test Data Selection. in: Proceedings of the 6th international conference on Software engineering. Tokyo, Japan. 1982. CA USA: IEEE Computer Society Press, 1982. 272~278

[28] B. Korel, A. M. A1-Yami. Assertion-Oriented Automated Test Data generation. in: Software Engineering, 1996., Proceedings of the 18th International Conference on , 25-30 March 1996. Berlin, Germany. 1996. New York, NY USA: ACM Press, 1996. 71~80

[29] A. Gotlieb, B. Botella, M. Rueher. Automatic Test Data Generation using Constraint Solving Techniques. ACM SIGSOFT Software Engineering Notes, 1998, 23(2): 53~62

[30] 王忐言，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

[31]P. Stocks and D. Carrington. A Framework for Specication-based Testing. IEEE Transactions on Software Engineering, 1996, 22(11): 777~793

[32]James.A, Whittaker. 实用软件测试指南. 电子工业出版社，2003

[33](美)Pressman, R. S.著. 软件工程—实践者的研究方法. 机械工业出版社，1999

[34](美)Charles F. Goldfarb, Paul Prescod 著. XML 实用技术. 清华大学出版社，1999.9

[35]http://www.w3c.org

[36]http://www.bearcave.com/software/antlr

[37] (美）李(Li,K.)等著；曹文静，谈利群等译. 高效软件测试自动化[M]. 北京：电子工业出版社，2004

[38]郭聪宾.利用 ANTLR 生成 C++描述的分析程序.程序员.2004.07: 103~106.

[39]蔡忠贤. 软件自动化测试的研究与实践. 南京理工大学硕士学位论文. 2005.

在读期间科研成果简介

1、论文《UML 在实时系统测试环境建模中的应用》发表于四川大学学报(自然科学版)2005 年.42(6).

2、论文《UML 在实时系统模拟测试数据程序开发中的应用》发表于四川师范大学学报(自然科学版)2005 年.增刊

声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

指导教师：[illegible]

学生：[illegible]

2006.5.27

致谢

本文的工作从研究方向的确定、论文的选题到定稿都是在导师洪玫副教授的悉心指导下完成的。在四川大学三年硕士的学习和生活中，自始至终得到了洪老师悉心的关怀和指导。导师渊博的知识、严谨的治学态度、勤恳的工作态度都使我受益匪浅。在研究生生活即将结束之际，谨对导师多年的辛勤培养和关心表示衷心的感谢，并向她表示深深的敬意。

我还要深深地感谢费向东老师以及 SDP 组的同事们，无论是工作上还是学习上他们都给了我不懈的支持和鼓励，并给我提出了许多建设性的意见和宝贵的建议，启发我的思维，使我能更深刻地考虑问题。

感谢川大智胜公司给我提供了实习的机会。感谢 ATC 项目组的同事们，他们在我参与项目以来，也给予了我无私的支持和帮助。

感谢我的同学们，感谢我的室友，在这三年中得到了他们的很多关心和支持与信任，也结下了深厚的友谊。我为能在这个充满进取精神的团结的集体中学习和工作，与各位老师、同事和同学愉快的合作感到由衷的庆幸。

最后，我要特别感谢我的丈夫和女儿。他们一直给我精神上的鼓励、物质上的支持、学习上的帮助和督促，用言辞无法表达我对他们的感激之情。他们的关怀和鼓励，是我前进的最大动力。

ATC系统软件自动化单元测试工具的研究与实现

作者： 张秀琼
学位授予单位： 四川大学