软件工程案例教程：基于路径测试技术的测试工具案例

郁 莲 编著

高等教育出版社

软件工程案例教程：
基于路径测试技术的测试工具案例

郁 莲 编著

RUANJIAN GONGCHENG ANLI JIAOCHENG :
JIYU LUJING CESHI JISHU DE CESHI GONGJU ANLI

高等教育出版社·北京

内容提要

本书配合软件案例驱动教学模式，以教师为主导、学生为主体，通过“基于路径测试技术的测试工具案例”，使学生深入理解和掌握该案例本身所反映的软件测试基本原理、技术和方法，及其他相关的软件工程知识，进而提高分析问题和解决问题的能力，最后通过案例的完成掌握软件工程实践方法，提高软件工程综合实践能力。本书适用作为软件开发及质量保证方向的高年级本科生及研究生的案例教材，也可作为广大软件工程实践者的参考资料。

图书在版编目（CIP）数据

软件工程案例教程：基于路径测试技术的测试工具案例／郁莲编著. --北京：高等教育出版社，2015.7

ISBN 978-7-04-033658-0

Ⅰ. ①软… Ⅱ. ①郁… Ⅲ. ①软件工程-案例-高等学校-教材 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2014)第 039961 号

策划编辑 倪文慧	责任编辑 倪文慧	封面设计 赵 阳	版式设计 于 婕
插图绘制 尹 莉	责任校对 殷 然	责任印制 尤 静	

出版发行	高等教育出版社	咨询电话	400-810-0598
社 址	北京市西城区德外大街 4 号	网 址	http://www.hep.edu.cn
邮政编码	100120		http://www.hep.com.cn
印 刷	大厂益利印刷有限公司	网上订购	http://www.landraco.com
开 本	787 mm × 1092 mm 1/16		http://www.landraco.com.cn
印 张	25.75	版 次	2015 年 7 月第 1 版
字 数	560 千字	印 次	2015 年 7 月第 1 次印刷
购书热线	010-58581118	定 价	42.00 元

物 料 号 33658-00

前　言

本书的编写目的：

基于案例的教学方法最早来源于法学和医学，后来又被广泛应用于工商管理类课程的教学中。为推动工程实践教学改革，进一步提高研究生工程实践教学质量和培养质量，北京大学软件与微电子学院软件技术与服务工程领域学科组认真规划、精心组织，鼓励和支持学院教师积极参与软件案例项目的开发和案例库建设，并在此基础上不断探索软件工程案例教学，以适应软件工程实践教学改革要求和人才培养需要。

本书配合软件案例驱动教学模式，以教师为主导、学生为主体，通过“基于路径测试技术的测试工具案例”，使学生深入理解和掌握该案例本身所反映的软件测试基本原理、技术和方法，及其他相关的软件工程知识，进而提高分析问题和解决问题的能力，最后通过案例的完成掌握软件工程实践方法，提高软件工程综合实践能力。

本书包括4部分内容。

第一部分为“导引”，简要介绍案例背景、案例学习目标、预备知识和软件项目的背景、任务、目标等。

第二部分为“相关知识”，介绍案例项目开发所涉及的软件工程相关知识、平台相关知识、领域相关知识等，这部分内容主要给出了案例相关的知识导图，并对案例所涉及的关键知识点进行简要介绍，旨在在案例学习过程中帮助读者梳理相关知识点。

第三部分为“案例实践”，按照软件工程过程组织该部分内容，包括项目计划、软件需求、软件设计、软件实现、软件测试、系统交付等环节。这部分内容包含案例开发各个阶段的关键方法和技术的讲解，并安排有启发式、引导式、讨论式案例学习的章节。例如，“方法指导”小节中给出系统设计及实现的框架分析、代码分析、测试分析等方法，既可以方便学习者预习，也可以作为教师引导式教学的内容，并和第二部分“相关知识”一起构成理论教学部分；“实践指导及总结”小节则是给学生提出课后实践的要求及内容、提出启发性问题等。

第四部分为“软件项目管理”，本书中的案例为中型软件案例，项目团队一般为6~10人，项目研发周期为3~4个月，为保证软件项目质量，在项目研发过程中需要按照项目管理方法组织和控制开发过程。这部分内容主要介绍软件开发过程中的项目管理方法和实践，不同的案例采用不同的项目管理方法和手段，一般包括项目启动、项目计划与执行、项目监督与控制等内容。通过这部分内容的展示和实践，使学生了解项目管理方法，通过实践环节参与到项目管理的过程中，进而体验软件项目管理的全过程和重要性。

课程或学时安排：

本书可以作为独立课程教材使用，也可以配合软件工程、软件测试等相关课程作为

实验教材使用。课程或学时安排具有灵活性，以下所述仅是一种案例教学建议。

第一部分“导引”内容简单，建议安排学生自学1个学时。第二部分“相关知识”，根据学生掌握案例项目开发所涉及的软件工程相关知识、平台相关知识、领域相关知识等情况，建议安排学生自学和教师辅导2~4个学时。第三部分“案例实践”涵盖本教程的主体内容，共6章（第六章至第十一章），建议每章安排2个学时。第四部分“软件项目管理”，建议安排学生自学2个学时，并参考本书中的案例项目团队组织方式建立自己的课程项目团队。

软件工程的案例驱动教学实践、教学改革还处于不断地探索和研讨中。限于作者的水平和能力，本书对于案例某些方面的描述可能不透彻，甚至可能有错误的地方，恳请读者批评指正。

本书的配套资源：

本书的配套教学资源包括电子教案、案例代码等，均可从中国高校计算机课程网下载。网址 http://computer.cncourse.com。

作者

2014年6月

目　录

第一部分 导 引

软件系统的开发包含了一系列的生产活动，人是这些活动的主体，在其中扮演重要角色的同时，难免会引入许多错误因素。这些因素可能出现在开发活动的最初，也可能出现在后期的设计和实现阶段，但却无一例外地影响着软件系统的质量。软件测试作为保证软件产品质量的重要手段之一，贯穿于软件系统开发过程的始终。

软件测试按照所使用的技术可以分为白盒测试、黑盒测试和灰盒测试。其中，白盒测试也称结构测试或逻辑驱动测试，它需要了解被测对象的内部工作，关注程序的结构和内部逻辑，并据此设计用例、测试程序，通过测试来检测被测对象内部动作是否按照设计规格说明书的规定正常进行，检验程序中感兴趣的每条通路是否都能按预定要求正确工作。

白盒测试有静态和动态之分，测试方法有多种，其中运用较为广泛的是基本路径测试法。此外，从覆盖标准角度而言，条件测试也是白盒测试的重要组成部分之一。条件测试的测试标准主要有条件覆盖策略、判定覆盖策略和多条件覆盖策略。

随着软件测试在软件开发过程中的重要性日益凸显，白盒测试也因其广泛的应用而备受关注。然而迄今为止，白盒测试仍以手工编写测试用例为主，市场对于自动化测试工具的需求与日俱增。基于这一现状，我们从教学和实践案例研究的角度出发迭代开发了一个案例测试软件：白盒测试平台（White Box Testing Platform），该平台基于基本路径测试法，兼有条件测试的相关标准实现，力求使学生能够从理论和实践的结合中切实了解测试对于软件质量保证的重要性，掌握白盒测试技术的自动化基本方法和流程，最大程度地减轻手工作业负担。

本部分内容旨在对该平台作简要的陈述，包括第一章“引言”和第二章“项目简介”。其中，“引言”部分从案例背景、学习实践目标和预备知识等方面介绍该平台涉及的测试技术、实现功能、应用技术、实践目的以及操作该平台之前所需具备的语言基础；“项目简介”部分通过介绍项目背景、项目任务和项目目标，对该平台进行概括性的描述。

第一章 引言

作为开篇章节，本章的主要内容包括软件测试案例的背景、学习实践目标以及学习前需要了解的预备知识。

1.1 案例背景

测试是保证软件质量的重要手段。IEEE 标准 610.12 给出了测试的两个定义：①测试是在特定的条件下运行系统或构件，观察并记录结果，对系统的某个方面做出评价；②分析某个软件项以发现现存的和要求的条件之差别（即错误）并评价该软件项的特性。

测试目的不仅仅是为了发现软件缺陷与错误，也是对软件质量进行度量和评估，以提高软件的质量。测试要以最少的人力、物力和时间找出软件中的各种错误与缺陷，通过修正各种错误与缺陷提高软件质量，避免软件发布后由于潜在的软件缺陷和错误造成的隐患所带来的经济风险。同时，测试是以评价一个程序或者系统属性为目标的活动，测试是对软件质量的度量与评价，以验证软件的质量满足用户的需求的程度，为用户选择与接受软件提供有力的依据。

软件测试的重要性及其对软件质量优劣的意义是非常重要的。软件系统的开发包括一系列生产活动，其中由人带来的错误因素非常多。错误可能出现在程序的最初，当时的目标可能是错误的或描述不完整；也可能在后期的设计和开发阶段，因为人们不能完好无缺地工作和交流，软件开发过程中必须伴有质量保证活动。

在软件测试的任务和重要性被明确之后，软件工程领域开始投入大量的时间进行软件测试的研究。软件测试的分类具体见表 1-1 所示。

表 1-1 软件测试的分类

软件测试	按阶段划分	单元测试
		集成测试
		系统测试
		验收测试

续表

软件测试	按测试主体划分	开发方测试
		用户测试
		第三方测试
	按执行状态划分	静态测试
		动态测试
	按所用技术划分	白盒测试
		黑盒测试
		灰盒测试
	其他	回归测试
		冒烟测试
		随机测试

以上针对不同的分类标准将软件测试划分为不同的类型，而本案例关注的核心类型是白盒测试。

白盒测试有静态和动态之分，静态分析容易理解，是检查程序本身的逻辑、编码风格等；而动态测试则要求按一定的步骤动态运行待测程序，在运行的过程中进行监测，测试平台必须对多种白盒测试技术的运行监测方式提供统一而完整的支持。

白盒测试的测试方法有代码审查法、静态结构分析法、静态质量度量法、逻辑覆盖法、基本路径测试法、域测试、符号测试、Z 路径覆盖和程序变异。其中运用较为广泛的是基本路径测试法。基本路径测试法是在程序控制流图的基础上，通过分析控制构造的环路复杂性导出基本独立路径集合，从而设计测试用例的方法。设计出的测试用例要保证在测试中程序的每个可执行语句至少执行一次。

此外，本案例平台还涉及条件测试。条件测试是检查程序模块中所包含逻辑条件的测试用例设计方法。条件测试的覆盖策略主要有 4 种：条件覆盖策略、判定覆盖策略、短路的多条件覆盖策略和非短路的多条件覆盖策略。

条件覆盖（condition coverage）是指设计若干测试用例，使程序中的每个条件的可能取值至少满足一次。对于每一个独立条件里的多个原子表达式，每个原子表达式必须先取一遍真值（true），再取一遍假值（false），从而生成满足条件覆盖策略的约束集。

判定覆盖（decision coverage）是指设计若干测试用例，使程序中的每个判断真假的分支至少遍历一次。因此在条件测试工具中，对于每一个独立条件，不管独立条件中有多少个原子表达式的组合，都先直接取该独立条件为真值（true），再取该独立条件为假值（false），从而生成满足判定覆盖策略的约束集。

多条件覆盖（multiple decision coverage）是指设计足够的测试用例，使得每个判定

条件中的条件的各种可能组合都至少出现一次，和条件覆盖一样，多条件覆盖不包括判定覆盖。在本白盒测试工具中，将多条件覆盖分为短路的多条件覆盖和非短路的多条件覆盖。对于 Visual Basic 和 Pascal 等不含有短路操作符（short circuit operators）的语言，多条件覆盖实际上是对于逻辑表达式的路径覆盖，和路径覆盖具有相同的优缺点。对于 C、C++和 Java 等具有短路操作符的语言，多条件覆盖所要求的一些组合测试在执行时无法达到。多条件覆盖所需要的测试用例的数目对于具有相似复杂性的条件却有非常大的不同。

以上简单介绍了白盒测试中基本路径测试和条件测试的相关概念，在实际进行白盒测试时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字，但即使每条路径都测试了仍然可能有错误。第一，穷举路径测试绝不能查出程序违反了设计规范，即程序本身是个错误的程序。第二，穷举路径测试不可能查出程序中因遗漏路径而出错。第三，穷举路径测试可能发现不了一些与数据相关的错误。

白盒测试工具必须能够完整地理解程序。理解程序的基础就是抽象语法树的获取，因此白盒测试平台必须对待测程序的语法提供完整的分析，提供完整的抽象语法树支持。

在抽象语法树的分析基础之上，还必须要提炼出整个程序框架的结构，如类列表、方法列表、变量表等。类和方法列表为自动化白盒测试提供基础（依次自动选择待测类、待测方法等），而变量表则是数据流分析的基础。

除了抽象语法树之外，白盒测试必须要对待测程序在运行测试用例的过程中进行监测。监测方式一般是对待测程序进行插桩监测。不同的测试需求以及不同的待测语言都会导致插桩方式的不一致。因此对多种插桩方式必须提供多种支持。

在白盒测试领域中目前主要以人工编写测试用例为主。使用一个能够生成测试用例的工具，对白盒测试本身是一个有效的改善途径；而使用基于工作流的测试应用将使这个平台可以方便地扩展功能。

原有的白盒测试工具由于没有统一底层的支撑、统一的插桩，编译及运行监测机制，加上部分工具经过多次多人迭代开发，使得维护工作难以继续进行。

本白盒测试平台设计了基本路径测试的插桩逻辑，对 break、continue、return、try…catch 等节点的插桩监测；改进了寻路算法；新增了静态代码监测的机制；封装了遗传算法来自动生成测试用例。

此外，本白盒测试平台底层还提供了对测试步骤的工作流的支持，并且工作流之间还能够组合。有了工作流的支持，白盒测试用户接口变得十分简洁；XML 格式的测试结果的输出，也为进一步封装为 Web 服务并为以 Web 服务方式提供测试奠定了基础。

多种插桩风格的支持以及变量表的提供也为以后的开发人员新增语句覆盖率测试，以及实现数据流测试提供了便利。开发人员可以参考现有的白盒测试应用的实现，快速地完成满足新需求的白盒测试应用。

本白盒测试平台的执行可分为4个阶段：解析Java源程序生成抽象语法树（AST）、遍历AST生成流图并插桩代码、导出测试用例和自动执行测试。下面简要介绍有关AST、用于生成测试用例数据的遗传算法以及执行测试的平台框架。

1. 抽象语法树（AST）背景知识

白盒测试中的静态分析主要以编译器来对源码进行解析，通过静态分析技术可以发现编程语言的错误，并且以故障或者警告的方式进行报告。当前，一种比较流行的方式就是通过JavaCC编写源码的编译器。

Java Compiler Compiler（以下简称JavaCC）是一个当时由SUN公司（现为Oracle公司收购）开发的，使用Java作为开发语言的语法分析生成器。这个分析生成器工具可以读取上下文无关且有特殊意义的语法，把它转换成可以识别且匹配该语法的Java程序，并且生成相应的抽象语法树。

通过词法分析与语法分析可以获取软件组成的重要基本因素，如变量标识符、过程标识符、常量等，通过组合这些基本因素可以得到软件的基本信息。

JavaCC使用被称为.jj的文件，该文件中的语法描述是使用非常类似于BNF的表示法编写的，这样从一种形式转换到另一种形式通常就相当容易（该表示法有自己的语法，从而使其在JavaCC中是可表达的）。JavaCC的.jj文件语法和标准的BNF之间的主要区别在于：利用JavaCC版本可以在语法中嵌入操作。一旦成功遍历了语法中的操作部分，则执行这些操作。操作都是Java语句，它们是解析器Java源代码的一部分，该部分作为解析器生成过程的一部分产生。

BNF是指巴科斯范式（Backus-Naur Form，BNF），是由John Backus和Peter Naur首先引入的用来描述计算机语言语法的符号集。现在，几乎每一位写编程语言书籍的作者都使用巴科斯范式来定义编程语言的语法规则。

例如，

> 在双引号中的字（“word”）代表这些字符本身，而double_quote用来代表双引号。在双引号外的字（有可能有下划线）代表语法部分。
> 尖括号（<>）内包含的为必选项。
> 方括号（[]）内包含的为可选项。
> 大括号（{}）内包含的为可重复0至无数次的项。
> 竖线（|）表示在其左右两边任选一项，相当于“OR”的意思。
> ::=是“被定义为”的意思。

.jj文件的编写可以使用LL(n)语法分析。通常JavaCC提供的默认方法是LL(1)词法分析，但是在特定情况下可以通过设定选项来指定使用LL(n)的语法分析，如LL(2)或者LL(3)语法分析；在分析完成后继续使用LL(1)分析。

下面是一个简单的支持“+”“-”号的加减法输入。

```
1  PARSER_BEGIN(Parser_1)
2  package examples.example_1;
3  public class Parser_1{}
4  PARSER_END(Parser_1)
5  void simpleLang():{}
6       {integerLiteral()(("+"|"-")intergerLiteral())? <EOF>}
7  void integerLiteral():{Token t;}{
8       t=<INT>{System.out.println("integer="+t.image);}}
9  SKIP:{""|"\t"|"\n"|"\r"}
10  Token:{<INT:(["0"-"9"])+>}
```

通过 Java 语言 BNF 编写的 .jj 文件作为输入后，就能生成对应 Java 源文件的基于 Java 语言的解析器（JavaParser）。通过这个 JavaParser 就能够将输入的源文件解析成抽象语法树（AST）。流程如图 1-1 所示。

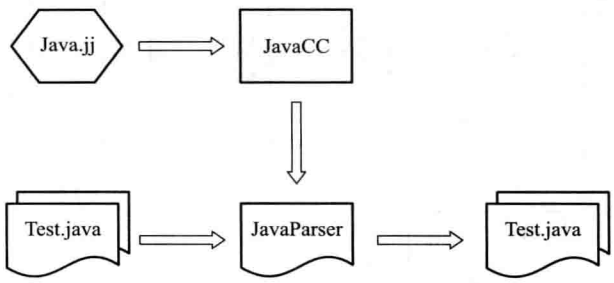

图 1-1　JavaCC 使用流程

有了 AST，就可以对符号表、控制流等进行进一步的分析了。如果语法本身有问题，那么一个源文件是无法成功产生 AST 的。AST 和 XML 文件的 DOM 树类似，AST 允许修改树模型并且可以把对模型的修改映射到 Java 源代码上。

2. 遗传算法背景知识

遗传算法（Genetic Algorithm，GA）是模拟达尔文生物进化论的自然选择和遗传学机理的生物进化过程的计算模型，是一种通过模拟自然进化过程搜索最优解的方法，它最初是由美国密歇根大学的 J. Holland 教授于 1975 年提出的，并出版了颇有影响的专著——《自然和人工系统的适应性》（Adaptation in Natural and Artificial Systems）。

遗传算法是从代表问题可能潜在的解集的一个种群（population）开始的，而一个种群则由经过基因（gene）编码的一定数目的个体（individual）组成。每个个体实际上是染色体（chromosome）带有特征的实体。染色体作为遗传物质的主要载体，即多个基因的集合，其内部表现（即基因型）是某种基因组合，它决定了个体形状的外部表现，如黑头发的特征是由染色体中控制这一特征的某种基因组合决定的。因此，在一开

始需要实现从表现型到基因型的映射即编码工作。由于仿照基因编码的工作很复杂，因此往往对其进行简化，如二进制编码，初代种群产生之后，按照适者生存和优胜劣汰的原理，逐代（generation）演化产生越来越好的近似解，在每一代，根据问题域中个体的适应度（fitness）大小选择（selection）个体，并借助于自然遗传学的遗传算子（genetic operators）进行组合交叉（crossover）和变异（mutation），产生代表新的解集的种群。这个过程将导致种群像自然进化一样的后生代种群比前代更加适应于环境，末代种群中的最优个体经过解码（decoding）可以作为问题近似最优解。

遗传算法作为一种高效的搜索寻优算法，在传统的程序直接执行技术中，为解决测试数据自动生成问题开阔了视野，提高了其实用化程度，在解决大空间、多峰值、非线性、全局优化等高复杂度问题时显示了独特的优势和高效性。因此，选用遗传算法生成测试数据。其工作流程如图 1-2 所示。

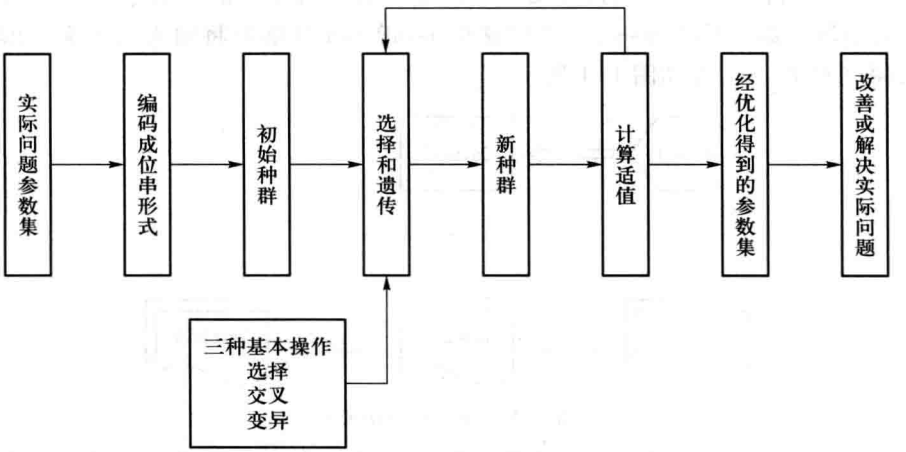

图 1-2　遗传算法工作流程

白盒测试平台的主要功能是针对 Java 源代码被测单元进行基本路径测试，包括生成测试路径集、生成测试用例、测试用例自动执行等子功能。首先求出程序或过程设计的逻辑复杂性测度，并以该测度为指南定义测试的路径集，生成测试用例并执行以达到某种测试的覆盖标准。

3. TaaS 框架

由于本系统是基于 TaaS（Testing as a Service）测试服务框架开发的，在这个框架下要把需要实现的功能按照一定的服务接口发布成 Web 服务，然后利用该框架的上层接口来调用服务即可。设计这个子系统时，先把它分成几个低耦合的模块，然后再把这些模块发布成服务，上层采用一定的流程执行语言来调用。

该测试平台计划发行两个版本，一个版本是 Web 应用形式，一个版本是 Eclipse 插件形式。使用 Web 版本，用户可以利用浏览器把被测的 Java 源代码上传到 Web 服务器，并

选择被测类和被测方法，服务器端生成相应的测试用例及执行结果并返回给用户。

在本测试平台的 Web 应用版本中使用 Flex 作为前台页面，系统结构如图 1-3 所示。

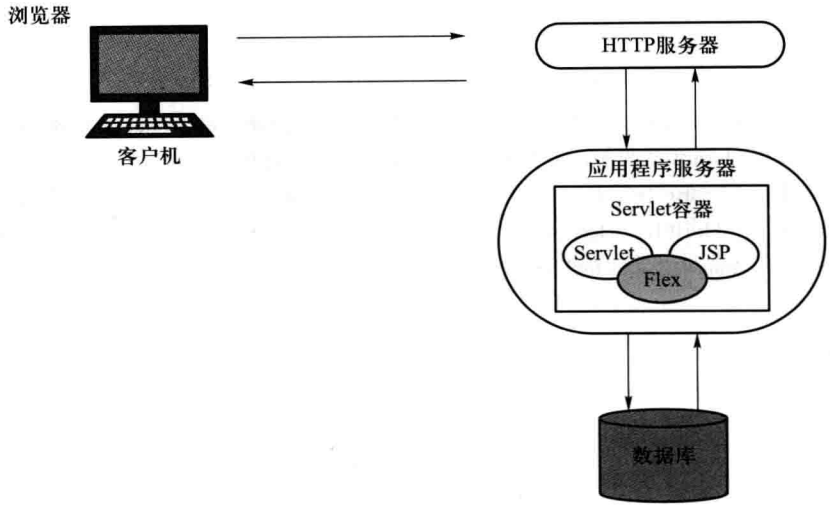

图 1-3　白盒测试平台的 Web 应用版本

为了支持分布式的持续开发集成，本白盒测试平台还提供 Eclipse 插件版本，它将平台所提供的功能封装成 Web 服务，供远程用户使用，且使其方便地与其他不同平台的应用程序进行协作。客户端采用 Eclipse Plug-in 的形式，这样用户在使用 Eclipse 进行编码的时候，只需要在 Eclipse 的相应菜单单击按钮即可完成对被测单元的白盒（基本路径和条件）测试。目前该版本仍在开发中，故下面以 Web 应用版本为主进行介绍。

1.2　学习实践目标

（1）掌握软件测试技术的基本概念，主要了解基本路径测试的概念和满足相应覆盖率的测试方法。

（2）学习使用开源工具 JavaCC 生成 JavaParser，解析 Java 源代码。

（3）掌握 Web 服务技术，以及对 Web 服务进行发布和访问。

（4）深入学习遗传算法，掌握遗传算法生成测试用例数据的原理和过程，并在此基础上提出对遗传算法改进的方案以达到提高测试覆盖率的目的。

（5）掌握 Eclipse 插件开发技术，将白盒测试工具封装成可安装的 Eclipse 插件。

1.3　预备知识

在学习本案例之前，学习者需具备软件工程的基本思想，对软件开发过程有一定的

认识和理解，熟悉软件项目过程管理，具有一定的面向对象编程语言基础，如：Java、C++。

参考文献

[1] 廖兴，尹俊文，蔡放. 基于 Java 语言的抽象语法树的创建与遍历. 长沙大学学报，2004.

[2] ECKEL B. Java 编程思想. 陈昊鹏，译. 4 版. 北京：机械工业出版社，2007.

[3] BLOCH J. Effective Java（中文版）. 杨春花，俞黎敏，译. 2 版. 北京：机械工业出版社，2009.

[4] BERNDT D，FISHER J，Johnson L，et al. Breeding software test cases with genetic algorithms. Proc of the 36th Annual Hawaii Int'l lonf on System Science，2003：338.