2004年上海大学博士学位论文 14

基于面向对象形式规格说明的测试用例生成技术

作 者：刘 玲
专 业：控制理论与控制工程
导 师：缪淮扣

上海大学出版社
·上海·

2004 年上海大学博士学位论文

基于面向对象形式规格说明的测试用例生成技术

作　者：刘　玲
专　业：控制理论与控制工程
导　师：缪淮扣

上海大学出版社
· 上海 ·

Shanghai University Doctoral Dissertation (2004)

Object-oriented Formal Specification Based Test Generation

Candidate: Liu Ling
Major: Control Theory and Control Engineering
Supervisor: Prof. Miao Huaikou

Shanghai University Press
• Shanghai •

上 海 大 学

本论文经答辩委员会全体委员审查，确认符合上海大学博士学位论文质量要求。

答辩委员会名单：

主任：	孙永强	教授，上海交大	200030
委员：	钱乐秋	教授，复旦大学	200433
	陈涵生	研究员，华东计算所	200233
	绍志清	教授，华东理工大学	200237
	吴耿峰	教授，上海大学	200072
导师：	缪淮扣	教授，上海大学	200072

评阅人名单：

钱乐秋	教授，	复旦大学	200433
郑国梁	教授，	南京大学	210093
刘宗田	教授，	上海大学	200072

评议人名单：

绍志清	教授，	华东理工大学	200237
李明禄	教授，	上海交通大学	200030
蒋昌俊	教授，	同济大学	200092
童维勤	教授，	上海大学	200072

答辩委员会对论文的评语

论文选题具有理论意义和应用价值。

该论文的研究成果在以下几点上有所创新:

(1) 提出了一个对逻辑覆盖测试准则进行评估的公理化系统;

(2) 提出了一组类内部测试准则;

(3) 提出了一系列用于测试类间多态关系的测试准则;

(4) 提出了一个描述测试用例推导过程的测试框架;

(5) 给出了一种应用线性规划和非线性不等式组球界具体测试数据的方法。

论文反映出作者具有坚实宽广的理论基础和系统深入的专门知识,独立开展科学研究工作能力强。论文结构合理,内容丰富,论述清楚,叙述流畅。在答辩过程中,回答问题思路清晰,能予以正确回答。

答辩委员会表决结果

经答辩委员会表决，全票同意通过张颖同学的博士学位论文答辩，建议授予工学博士学位。

答辩委员会主席：孙永强

2004 年 3 月 11 日

摘　要

随着信息技术的发展，软件的规模不断扩大，如何保证和提高软件质量成为软件界最为关心的问题之一. 软件测试作为保证软件质量的关键技术之一，能够有效地发现软件中的故障. 据统计，在软件开发成本中，软件测试的工作量往往占软件开发总工作量的 40%以上. 对于某些安全关键的软件，其测试费用甚至高达所有其它软件工程阶段费用总和的三到五倍. 因而，提高软件测试的有效性和测试效率、降低软件开发成本已成为软件工程师迫切需要解决的任务之一.

软件测试的核心问题是测试用例的产生. 根据测试用例产生的依据可以将软件测试分为基于程序代码的测试和基于规格说明的测试两大类. 基于程序代码的测试是指测试者根据程序的内部结构和与路径相关的数据特性设计测试用例. 基于规格说明的测试是指测试者不需了解程序的内部结构，直接根据程序的需求规格说明来确定测试用例，推断测试结果的正确性. 这种测试可以验证程序实现是否符合系统需求，从而实现软件测试的目的. 同时，由于测试用例的开发是以系统规格说明为依据，测试用例的设计过程可以和程序开发过程同时进行，缩短了软件开发周期，提高了软件开发的效率.

形式化的软件规格说明具有精确的符号表示和清晰的语义，能够进行推理和证明，同时也便于工具的处理. 因此，用形式化

的规格说明作为测试用例的来源将会使得测试用例的自动构造成为可能. 但目前基于形式规格说明构造测试用例的研究主要集中在利用黑盒测试准则（如：范畴划分测试准则，基于状态图的测试准则）从不同形式的规格说明（如：Z 模式、UML 状态图）中构造测试用例. 但是对于基于模型的面向对象形式规格说明（如：Object-Z、VDM++规格说明）应用这些构造测试用例的技术非常困难. 因为，这些基于模型的面向对象规格说明没有提供状态图，也很难将范畴划分测试准则应用到这些规格说明中用于定义输入空间的复杂谓词上.

论文提出了一系列用于从基于模型的面向对象形式规格说明推导测试用例的方法和技术. 这些方法和技术分别探讨了在类测试层次以及类间测试层次上如何从基于模型的形式规格说明构造测试用例. 在类测试层次上，论文定义了一个公理化系统对现有的可应用于方法内测试的逻辑覆盖准则进行评估，为测试人员从这组准则中选择某个准则提供了依据. 此外，在这一层次的测试过程中，论文还提出了一组用于产生测试类的方法间交互的类内部测试准则. 这组准则利用方法内测试用例来构造类内部测试用例，避免了以往的研究中采用基于状态图的测试准则构造测试用例时需要的抽取类的状态图的工作，使得机械化地构造类测试用例成为可能. 在类间测试层次，论文给出了一系列用于测试类间的多态关系的测试准则，改变了以往对多态关系的测试主要集中于基于程序代码的方法. 应用论文提出的这些用于测试类间多态关系的测试准则，测试人员可以从形式规格说明中推导出用于测试多态对象的测试用例，这些测试用例可以最终检测出程序代码中的多态关系的实现与规格说明中的定义是否一致.

除了给出若干个用于从基于模型的面向对象形式规格说明

构造测试用例的测试准则之外，论文还提出了一个用于描述这种测试用例推导过程的测试框架. 用形式规格说明语言 Object-Z 来定义论文提出的测试框架，测试框架的描述与规格说明的描述所采用的形式规格说明语言一致，从而避免了用另一种描述语言给读者带来的额外负担. 论文对测试框架的阐述包括两个部分. 第一部分是类测试框架. 它用来描述类测试用例的推导过程，其中包括方法内测试用例、类内部测试用例、方法内测试用例的产生过程以及类内部测试用例推导过程的定义. 第二部分包括如何利用类层次结构在子类中重用父类的类测试框架. 利用论文提出的这个测试框架，测试用例生成系统在构造测试用例的同时可以自动记录测试用例生成过程，从而为测试用例的管理和追踪提供了方便.

最后，论文给出了一种应用求解线性规划和非线性不等式组对测试规格说明中定义的测试输入进行求解得到可以运行的具体的测试数据的方法，并且介绍一个实现了上述提出的测试技术的系统原型.

关键词 基于规格说明的软件测试，测试准则，测试框架，面向对象测试，测试用例生成

Abstract

With the development of information technique, the software grows larger and larger. How to guarantee and improve the software quality becomes the main concerned point in the field. As one of the key techniques of guaranteeing software quality, software testing can effectively detect the faults in the system. According to the statistics, the cost of software testing accounts for 40% in the whole cost of software development. With the enlarging of software, the proportion of software testing grows larger in the software development. For some safety critical system, the cost of testing almost corresponds to the triple or quintuple cost of all other developing phase. Thus, improving the effectivity and efficiency of testing and reducing the software developing cost become one of the urgent tasks for software engineers.

The core problem of software testing is test cases generation. According to the source that is used to derive tests, software testing is classified into two categories – program based testing and specification based testing. Program based testing means testers use program structures to derive test cases. Specification based testing means testers use the specification instead of program structures to derive test cases. The specification based testing can verify whether a program conforms to its specification, which is the aim of testing. Furthermore, during specification based testing process, the

development of test cases can be parallel with the software development. That can reduce the time of software development, and improve development efficiency.

Since formal specifications possess precise notations and clear semantics, and can be used in reasoning and proving, taking formal specifications as the source of generating test cases makes automatic test cases generation possible. However, most research of formal specification based test generation focuses on applying black-box test criteria (such as category partition criteria, state graph based criteria) on different forms of formal specifications (such as, Z schema, UML state chart) to generate test cases. For model-based object-oriented formal specifications (such as, Object-Z, VDM++ specifications), those methods are very hard to be applied. The main causes are that there is no state chart notation in the specifications and it is hard to apply category partition criteria to those complicated predicates in the specifications.

The article provides a series of techniques that is used to derive test cases from model-based object-oriented formal specifications. Those techniques discuss the problems of deriving test cases at class level and intra-class level respectively. At class testing level, the article defines a set of axioms that is used to evaluate the existing logic coverage criteria, which are applied in the intra-method testing. The evaluation provides testers with a guide of selecting logic coverage criterion. Moreover, at the class testing level, the article also presents a set of intra-class test criteria, which is used to generate intra-class test cases from intra-method tests. Those

intra-class test cases can test the interactions among the public methods of a class. Those criteria avoid the extra burden of extracting class state graph from class specification and make mechanical generating intra-class test cases possible. At inter-class testing level, the article presents a series of test criteria that are used to test the polymorphism relation in a class hierarchy. These inter-class criteria are different from program based polymorphism test criteria. They are used to derive test cases from formal specification. These test cases can check whether the polymorphism relationship implemented in a program conforms to that defined in the specification.

Besides above test criteria, the article also gives a testing framework to describe the test case generation process. This testing framework is defined with Object-Z notation, which is adopted in the specifications. It reduces the understanding burden for readers while adopting another notation. The discussion of testing framework in the article includes two parts. One is the definition of class testing framework, which is used to describe the class tests generation process. It includes the descriptions of intra-method test case, intra-class test case, the deriving process of intra-method test cases and intra-class test cases. The other part specifies the techniques of reusing parent class testing framework in sub-class testing. With the testing framework presented in the article, the test case generation system can document the test generation process while producing test cases. Thus, test cases managing and tracking become more convenient.

Finally, the article introduces an approach to instantiating the test case specification into test data and a system prototype that implements above testing techniques.

Key words specification based software testing, test criteria, testing framework, object-oriented testing, test cases generation

目 录

第一章 绪 论

1.1 软件测试概述

在日常生活中，我们无时无刻不在和软件打交道. 当打电话时，我们在使用控制电话机交换的软件；当去银行取钱或存钱的时候，我们在使用银行的管理软件；当乘飞机的时候，我们要依赖飞机控制系统软件的精确控制. 现代社会中人们对软件的这种依赖性使得软件质量成为大家关注的焦点. 而对于失败将导致人员伤亡这类“安全至上”的系统（如空中交通管制系统、导弹制导系统、或医用输送系统）来说，软件的质量就显得更加至关重要. 因此，如何保证软件质量以及如何最大限度地提高软件质量成为摆在软件工程师面前的一个严峻的课题.

软件测试作为保证软件质量的重要手段从20世纪70年代以来逐渐受到人们的重视，也出现了很多有关软件测试的方法和实践的文章. 从这些文章中我们可以看出软件测试的主要目的如下：

- 核实软件的所有需求是否都已正确实施；
- 核实软件的所有构件是否都正确集成；
- 核实每个对象内部的交互是否正确；
- 确定缺陷并保证在部署软件之前将缺陷解决.

为了实现软件测试的上述目的，传统的软件测试过程遵循图