蜕变测试与适应性分区测试的集成方法与支持工具研究

代贺鹏

北京科技大学

密　　级：公开

论文题目：**蜕变测试与适应性分区测试的集成方法与支持工具研究**

学　　号：G20168664

作　　者：代贺鹏

专 业 名 称：软件工程

2018 年 5 月 4 日

蜕变测试与适应性分区测试的集成方法与支持工具研究

Research on Integration of Metamorphic Testing and Adaptive Partition Testing and Its Supporting Tool

研究生姓名：代贺鹏

指导教师姓名：孙昌爱

北京科技大学计算机与通信工程学院

北京 100083，中国

Master Degree Candidate： Dai Hepeng

Supervisor： Sun Chang-ai

School of Computer and Communication Engineering

University of Science and Technology Beijing

30 Xueyuan Road，Haidian District

Beijing 100083，P.R.CHINA

分类号：TP311　　密　　级：公开

ＵＤＣ：004.41　　单位代码：１０００８

北京科技大学硕士学位论文

论文题目：蜕变测试与适应性分区测试的集成方法与支持工具研究

作者：代贺鹏

指 导 教 师：孙昌爱　教授　**单位：**北京科技大学

指导小组成员：　　**单位：**

指导小组成员：　　**单位：**

论文提交日期：2018 年 5 月 4 日

学位授予单位：北 京 科 技 大 学

致　谢

时间如白驹过隙，研究生的学习生活马上就要结束。期间，在孙昌爱教授的指导下，我的知识水平与专业技能得到了长足地进步，开阔了科技视野，锻炼了口头与书面的表达能力。

首先，由衷感谢我的导师孙昌爱教授在我毕业设计期间耐心地指导与大力地支持。孙老师循序渐进地引导我接触这个领域、探索这个领域、发现这个领域存在的问题、研究解决的方式。在整个学习、研究过程中，我印象最深的是孙老师严谨的学术态度、积极的人生价值观以及宽厚的品质。孙老师是我人生的灯塔、学习的榜样！

然后，感谢所有给我授过课的老师，他们孜孜不倦地教育一届又一届的学子。从他们身上，我感受到了书卷的气息、大师的风采、严谨的治学态度。同时，师姐师兄以及师弟们给了我无私的帮助，我衷心地感谢他们。

父书空满筐，母线萦我襦。父亲母亲是我们最温暖的港湾，他们的爱是最无私的，不断地鼓励我继续前进。真心希望他们身体健康、笑口常开。

摘　要

蜕变测试首先识别蜕变关系，然后根据原始测试用例与识别的蜕变关系生成衍生测试用例，分别执行原始测试用例与衍生测试用例，依据蜕变关系验证执行的结果。目前的蜕变测试方法中随机地执行测试用例，没有有效地利用测试过程信息。适应性分区测试在软件测试过程中引入反馈机制，提高了随机执行过程的故障检测效率。

本文旨在提高蜕变测试的故障检测效率，探索蜕变测试与适应性分区测试的集成方法，提出两种适应性蜕变测试技术，开发了相应的支持工具。具体说来，本文的主要研究工作包括：

(1) **提出了以蜕变关系为中心的适应性蜕变测试技术（M-AMT）：**M-AMT 在蜕变测试的基础上，引入反馈机制进行测试过程的控制，首先随机地选取蜕变关系，并执行相应的测试用例，依据当前测试结果更新测试剖面，然后依据测试剖面选择分区，在选取的分区中随机选择所有测试用例涉及的蜕变关系进行随后的测试过程。

(2) **提出了以分区为中心的适应性蜕变测试技术（P-AMT）：**P-AMT 在适应性分区测试的基础上，将蜕变关系作为一种验证测试结果的机制，首先依据测试剖面选择分区，在选取分区中随机地选择测试用例，然后依次根据与该测试用例相关的蜕变关系生成衍生测试用例，执行测试用例并依据测试结果更新测试剖面。

(3) **开发了适应性蜕变测试支持工具 APT2MT：**APT2MT 支持终止条件设置、分区信息设置、测试用例执行和测试报告打印等功能。借助该工具能进一步提高适应性蜕变测试技术的自动化程度。

(4) **采用经验研究验证了两种适应性蜕变测试技术的有效性与效率：**利用三个 Java 程序实例评估并比较了两种适应性蜕变测试技术和传统的蜕变测试技术的故障检测效率，以及选择测试用例的时间开销。

本文提出的两种蜕变测试与适应性分区测试的集成方法从不同角度改进了蜕变测试与适应性分区测试的局限性，M-AMT 提高了蜕变测试的故障检测效率，P-AMT 提高了适应性分区测试在测试预期不存在情形下的适用性，开发的支持工具提高了适应性蜕变测试技术的自动化程度。

关键词：　蜕变测试，适应性分区测试，测试预期，测试工具

Research on Integration of Metamorphic Testing and Adaptive Partition Testing and Its Supporting Tool

Abstract

Metamorphic testing (MT) first defines metamorphic relationships (MRs) which are used to generate new test cases (i.e. follow-up test cases) from available original test cases (i.e. source tests cases). Both source and follow-up test cases are executed and their results are verified against the relevant MRs. Up to now, most of MT techiniques randomly select test cases for execution, which does not use of any information returned from the testing process. Adaptive partition testing (APT) introduces a feedback mechanism to random testing, with an aim of improving the fault decetion efficiency of random testing.

This thesis aims to improve the fault detection efficiency of MT through exploring effective combinations of MT and APT, and accordingly, we have proposed two adptive metamorphic testing techiniques (AMT), and developed a corresponding supporting tool for them. Main contributions made in this thesis are as follows:

（1） An MRs-centric adaptive metamorphic testing technique (M-AMT): Based on MT, M-AMT adds a feedback mechanism to control the execution process of MT. First, M-AMT randomly selects an MR to generate source and follow-up test cases of related input partitions, and then updates the test profile of input partitions according to the result of test execution. Second, a partition is selected according to updated test profile, and an MR is randomly selected from the set of MRs whose source test cases belong to selected partition.

（2） A partition-centric adaptive metamorphic testing technique (P-AMT): Based on APT, P-AMT leverages MRs as a mechanism for verifying the test results. First, P-AMT selectes a partition according to the test profile. Second, a test case is randomly selected in the selected partition, its follow-up test cases are generated based on the involved MRs, and their results are verified against the involved MRs. Thrid, P-AMT updates the test profile based on the test results.

（3） A supporting tool of adaptive metamorphic testing called APT2MT has been developed to further improve the automation of the proposed adaptive metamorphic testing techniques, which has features such as termination condition setting, partition setting, test execution, and test report generation.

（4） An empirical study that has been conducted to evalue the fault detection effictiveness the efficiency of the proposed adptive metamorphic testing

techiniques, where three Java programs are selected as subjects to compare the performance of traditional MT and proposed adptive metamorphic testing techiniques in terms of fault detection effectivenss and time overhead in test case selection.

To sum up, the combination mtehods of MT and APT proposed in this thesis overcome the limitations of proposes two integration methods of MT and APT from different perspectives, namely, M-AMT improves the efficiency of fault detection efficiency of MT, while P-AMT improves the applicability of APT in situations where test oracle does not exist. The supporting tool further increases the automation of adaptive metamorphic testing techniques.

Key Words： Metamorphic Testing, Adaptive Partition Testing, Test Oracle, Testing Tool

目　　录

1 引言

本章介绍课题研究背景与意义、研究内容与成果，以及论文组织结构。

1.1 研究背景与意义

软件测试是一种广泛采用的软件质量保证手段[1]，通过运行有限的测试用例，比较测试用例的输出与预期是否一致来检测软件中潜藏的故障。然而在实际测试中，测试者构造测试预期可能需要花费相当大的测试资源或者无法构造测试预期。这类情况在软件测试中被称为测试预期问题[2,3]。测试预期问题限制了许多测试方法的实际应用[4]。

蜕变测试是 1998 年 T. Y. Chen 提出的一种缓解测试预期问题的测试技术[5]。该技术依据待测软件的蜕变属性获取蜕变关系，执行原始测试用例（采用测试用例生成技术获得）与衍生测试用例（根据蜕变关系获得），检查对应的输出是否违反蜕变关系。如果违反了某种蜕变关系，则表明软件中存在故障。蜕变测试无需测试预期并且简单易用，近年来受到广泛关注。

课题组将控制论[6]引入软件测试中并提出适应性分区测试技术[7]。适应性分区测试技术在测试的过程中根据测试结果是否符合预期动态改变测试剖面，使得失效率高的分区被选择的概率大。当不存在测试预期时，提出的适应性分区测试技术的适用性受到限制。

蜕变测试的相关研究主要关注如下方面[8]：（a）蜕变测试的基本理论；（b）与其它测试技术结合；（c）将蜕变测试应用到其它领域；（d）开发自动化的蜕变测试工具。然而，蜕变测试基本理论的相关研究主要通过生成或选择“较好”的蜕变关系（测试用例的执行结果容易违反这种蜕变关系）提高蜕变测试的故障检测效率，忽略了原始测试用例对蜕变测试效率的影响。

为了在原始测试用例的生成和选择方面提高蜕变测试的故障检测效率以及解决缺少测试预期时适应性分区测试不能测试软件的问题，本文根据蜕变测试的基本原理与适应性分区测试的特点，探索蜕变测试与适应性分区测试的集成方法，提出了两种适应性蜕变测试技术，开发了相应的支持工具，采用经验研究的方式评估并比较了本文提出的两种适应性蜕变测试技术与随机测试技术的有效性和选择测试用例的时间开销。

1.2 研究内容与成果

本文结合蜕变测试的基本理论与适应性分区测试的特点，提出了两种适

应性蜕变测试技术，开发了相应的支持工具并进行实例验证。主要研究内容与成果如下：

(1) **提出两种适应性蜕变测试技术：**本文将适应性分区测试技术中的控制论思想引入到蜕变测试中，提出了两种适应性蜕变测试技术：（a）以蜕变关系为中心的适应性蜕变测试技术，旨在提高蜕变测试的故障检测效率，首先根据原始测试用例与衍生测试用例的测试结果是否违反蜕变关系更新测试剖面，然后依据测试剖面选择分区，最后将选中分区涉及到的蜕变关系作为下一个待验证的蜕变关系；（b）以分区为中心的适应性蜕变测试技术，将蜕变测试中的蜕变关系作为适应性分区测试中测试结果的判断机制，旨在解决缺少测试预期时适应性分区测试技术无法应用的问题，首先根据测试剖面选择分区，然后在选中的分区中随机选择原始测试用例，依据与选中的原始测试用例相关的蜕变关系生成衍生测试用例，最后执行原始测试用例与衍生测试用例，并根据执行的结果是否违反蜕变关系更新测试剖面。

(2) **开发了适应性蜕变测试的支持工具 APT2MT：**该工具支持测试者选择不同的测试技术（M-AMT、P-AMT 和 MT）测试待测程序、设置终止条件、设置分区、执行测试和打印测试报告。

(3) **对提出的两种适应性蜕变测试技术进行经验研究：**采用三个真实的 Java 程序实例验证并评估提出技术的有效性与支持工具的实用性。

本文研究工作得到国家自然科学基金（面上项目）“面向 SOA 软件的蜕变测试技术研究”（项目编号：61370061）的资助。

1.3 论文组织结构

本文的组织结构安排如下：

第一章， 引言，主要包括课题的研究背景与意义、研究内容与成果以及论文组织结构。

第二章， 背景介绍，主要包括软件测试相关概念和软件测试相关技术的国内外研究现状。

第三章， 深入讨论蜕变测试与适应性分区测试的集成方法。。

第四章， 讨论适应性蜕变测试支持工具的设计与实现，包括需求分析、系统设计与实现和系统演示。

第五章， 对三个真实程序进行实例研究，验证适应性蜕变测试技术的有效性与效率，主要包括研究问题阐述、实验对象的说明、实验设计以及实验结果分析。

第六章， 研究工作的总结以及未来的展望。

2 背景介绍

本章介绍蜕变测试和适应性分区测试的相关概念、基础理论以及国内外研究现状。

2.1 相关概念与技术

本节首先介绍蜕变测试的基本理论与相关概念，然后介绍随机测试和分区测试的概念与特点，最后介绍变异测试相关概念。

2.1.1 蜕变测试

软件测试包括测试用例生成、测试执行与测试结果判定。测试预期是判断软件输出是否正确的机制。一般而言，传统的软件测试技术假设测试预期存在，通过比较测试用例的实际输出与测试预期是否相等来检验待测程序。实际情况中，有时候并不能得到或者需要很大的代价才能得到测试预期，这就是所谓的测试预期问题[2,3]。

蜕变测试[5]是一种可以缓解测试预期问题的测试技术，基本原理如图 2-1。首先通过待测程序的蜕变属性生成测试用例，然后执行测试用例（原始测试用例和衍生测试用例），最后验证输出结果是否违反了蜕变关系。如果违反了蜕变关系，表明待测程序中存在故障。

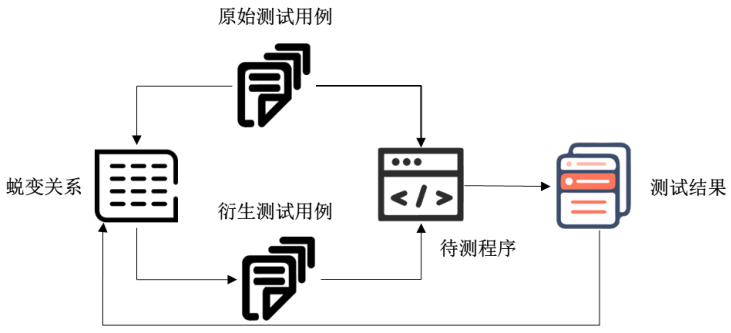


图 2-1 蜕变测试框架

定义 1（蜕变关系[9]）：假设函数f的程序实现是P，f的输入是$x_1, x_2, \ldots, x_n (\mathrm{n} > 1)$，对应的输出为$f(x_1), f(x_2), \ldots, f(x_n)$。如果输入序列$x_1, x_2, \ldots, x_n$满足关系$\mathrm{r}(x_1, x_2, \ldots, x_n)$，输出序列$f(x_1), f(x_2), \ldots, f(x_n)$满足$r_f(f(x_1), f(x_2), \ldots, f(x_n))$，由此可得：

$$\mathrm{r}(x_1, x_2, \ldots, x_n) => \ r_f(f(x_1), f(x_2), \ldots, f(x_n))$$

(r, r_f)称为程序P的一个蜕变关系。

在蜕变测试中，原有的测试用例（由某种测试用例生成技术产生）称为原始测试用例，依据蜕变关系产生的测试用例称为衍生测试用例。例如：对于一个计算 cos 函数值的程序P_{cos}，cos 函数具有$\cos x = \cos(x + 2\pi)$的性质。该性质可以作为一个蜕变关系。对于给定的两个输入x和$\mathrm{y} = x + 2\pi$，如果$\cos x \neq \cos y$，表明P_{cos}中存在故障。

2.1.2 随机测试与分区测试

随机测试（RT）[10]和分区测试（PT）[11]是两个主要的软件测试方法。随机测试选取测试用例的方式有两种：（a）按照均等的概率挑选测试用例；（b）根据功能剖面在输入域中随机地选择测试用例。功能剖面是根据待测软件不同输入数据的使用频率为测试用例设定选取概率：使用频率高的输入数据对应的测试用例具有较高的选取概率，使用频率低的输入数据对应的测试用例具有较低的选取概率。随机测试选取测试用例的方式简单并且测试人员易于使用该技术。然而没有利用待测软件的信息与执行的过程信息，随机测试的测试效率可能不高。

与随机测试不同，分区测试旨在同构地产生测试用例。分区的同构性是指相同分区的测试用例具有相似的软件行为，即如果分区中的一个测试用例揭示/没有揭示待测软件的故障，那么相同分区的其它测试用例也揭示/没有揭示待测软件的故障。分区测试首先将软件的输入域划分为若干不相交的分区，然后从每一个分区中选取测试用例并执行。

研究人员对 RT 和 PT 的故障检测效率进行了大量的对比[12-15]。虽然 PT 可以更系统地产生测试用例，但是 PT 并没有大幅度地改进 RT。甚至在一些情况下，PT 不如 RT。Chen 等人理论地分析了 PT 和 RT 的测试效率，发现只有每个分区选取的测试用例与分区的大小成比例时，PT 的测试效率在理论上高于 RT[4]。然而在实际情况下，分区的同构性质以及等比例的选取测试用例很难保证，因此 PT 的检测效率可能比 RT 低。

RT 与 PT 对不同类型故障的敏感程度不同，因此集成两种技术是一种自然的想法。Cai 等人将 RT 与分区测试技术集成并提出随机分区测试（RPT）[16,17]，首先将软件的输入域划分为m不相交的分区$c_1, c_2, \ldots, c_m$，然后等概率地选取分区c_i，最后在c_i中随机选取测试用例并执行。

2.1.3 变异测试

变异分析是一种基于故障的测试技术，运用变异算子模拟程序中常见的故障并将故障植入原始程序[18]。植入故障的程序称为变异体。测试人员使用测试用例分别在原始程序以及变异体上执行。若存在某一个测试用例的执行结果与原始程序不同，该变异体被“杀死”。任何测试用例都检测不出来的变异体称为等价变异体。使用某种测试用例生成技术产生测试用例集S，通过原始程序及变异体在S上的执行情况，评估产生测试用例集的充分性。在给定测试用例集的情况下，使用某种测试技术选择测试用例并分析原始程序及变异体的执行结果，可以评估测试技术的有效性。变异分析中最重要的评估指标是变异得分（MS），它是指测试用例集“杀死”的变异体数量占所有变异体的百分比，形式化定义如下：

$$\mathrm{MS}(\mathrm{P},\mathrm{S}) = \frac{N_k}{N_m - N_e} \tag{1}$$

在公式（1）中，P是原始程序，S是指某种技术产生的测试用例集，N_k表示被杀死的变异体数目，N_m表示变异体的总数量，N_e是指等价变异体的数目。

2.2 国内外研究现状

本节首先介绍蜕变测试与适应性分区测试国内外研究现状与进展，然后介绍课题组前期研究工作。

2.2.1 蜕变测试研究现状

蜕变测试无需测试预期，因此有效地缓解了测试预期问题，并在多个领域得到成功的运用[8]。近年来，研究者在蜕变测试理论与应用等方面取得了丰富的研究成果。部分代表性的工作如下：

（1） 蜕变测试理论研究

在蜕变关系识别与复合方面：Zhang 等提出一种基于搜索的蜕变关系推理方法[19]；Su 等人提出了一种蜕变关系动态识别方法[20]；Sun 等人提出一种数据变异指导的蜕变关系获取方法[21]；Gotlieb 等人提出了一种蜕变关系验证框架[22]，对于给定的程序的某个蜕变关系植入故障，利用约束逻辑编程技术生成满足植错后的蜕变关系的测试用例集，如果测试用例集检测出故障，则表明植错后的蜕变关系是错误的；Liu 等人提出一种蜕变关系复合方法，通过对多个蜕变关系的复合形成新的蜕变关系[23]。

METRIC 是一种基于范畴划分的蜕变关系识别方法[24]。该方法涉及的

相关概念包括范畴（Category）、选项（Choice）、测试帧（Test Frame）及完整测试帧（Complete Test Frame）[25]。

定义 2 (范畴) 程序参数或者环境的一个主要属性或者特征称为范畴。

定义 3 (选项) 与范畴相关联的各种可能的值被划分为互不相交的子集，这些子集称为选项。

定义 4 (测试帧) 不同范畴的选项组成的集合称为测试帧。

定义 5 (完整测试帧) 使程序正常执行的一组选项的集合称为完整测试帧。

测试人员首先根据规格说明书识别范畴及对应的选项，然后将不同范畴中的选项组合并形成完整测试帧。METRIC 选择两个有区别的完整测试帧，这两个完整测试帧具有一个选项或多个选项不同的特点，然后测试人员比较这两个完整测试帧并判断是否存在蜕变关系。METRIC 识别蜕变关系的框架包含一下三个步骤：

（a）两个不同有效的完整测试帧作为软件测试人员识别蜕变关系的候选对；

（b）测试人员判断候选对是否可以识别蜕变关系，如果可以识别蜕变关系，测试人员描述蜕变关系并记录；

（c）重复步骤（a）直到遍历所有的候选对或者识别蜕变关系的数目到达预先设定的值。

在蜕变测试原始测试用例生成方面：Batra 等人提出了一种基于遗传算法的测试用例生成方法，旨在最大化覆盖程序路径[26]；Dong 等人采用符号执行提取蜕变关系并生成相应的测试用例[27]；Chen 等人比较了特殊值法与随机方法产生的原始测试用例对蜕变测试效率的影响[28]。刘益强提出了一种基于符号执行的原始测试用例生成技术和一种基于路径距离的原始测试用例优先级排序方法，并开发了相应的支持工具[29]。

（2） 蜕变测试与其它测试技术结合

Xie 等人将蜕变测试与基于频谱分析的故障定位技术结合，提出了无需预期的故障定位方法[30]；Dong 等人将蜕变测试与遗传算法相结合，在搜索过程中使用蜕变关系，将原始测试用例和衍生测试用例都视为候选方案，加速达到某种覆盖标准[31]。Murphy 等人结合 Java 建模语言、断言技术以及蜕变测试技术成功地测试了存在测试预期问题的程序[32]。Liu 等人将蜕变测试技术与容错技术结合解决了某些容错技术缺少测试预期时不可用的问题[33]。

（3） 蜕变测试在不同领域中的应用

Sun 等人提出了面向 Web 服务的蜕变测试方法[34,35]，首先从 Web 服务的 WSDL 描述文档中提取蜕变关系，然后根据 WSDL 文档随机产生测试用例并且运用蜕变关系得到相应的衍生测试用例；Chan 等人提出了一种面向 SOA 的蜕变测试方法，将离线测试通过的测试用例作为原始测试用例，在线测试用例作为衍生测试用例[36,37]；Mayer 等人将蜕变测试应用于图像处理程序[38]；Kuo 等人将蜕变测试应用于图象处理程序时检测出一个真实故障[39]；Tse 等人尝试将蜕变测试应用于上下文敏感的中间件软件[40]；Chan 等人将蜕变测试应用于能量感知的无线传感器网络应用软件[41]；Sun 等人将蜕变测试成功地用于几类典型加密程序的测试[42]。

2.2.2 适应性分区测试研究现状

适应性分区测试技术（APT）是在动态随机测试技术（DRT）的基础上发展而来。Cai 等人将控制论引入到软件测试中，提出了利用测试过程的历史信息更新测试剖面的适应性测试技术[6]，但是适应性测试技术的时间开销太高。为了降低适应性测试技术的时间开销，Cai 等人根据引起故障的输入趋向于集簇在连续的区域这一观察[43,44]，提出了利用当前的测试信息更新测试剖面的动态随机测试技术（DRT）[45]。DRT 主要特点是在测试的过程中根据每一次的执行结果动态改变测试剖面，使得具有较高失效率的分区的选取概率变大：假设存在一个分区c_i，若该分区中的一个测试用例t揭示了软件的故障，c_i分区的选取概率变为$p_i+\varepsilon$。若测试用例t没有揭示软件的故障，c_i分区的选取概率变为$p_i-\varepsilon$，其中ε为调整分区选取概率的参数。DRT 更新测试剖面的具体步骤如下：

假设待测软件的输入域划分为m不相交的分区$c_1, c_2, \ldots, c_m$，每一个分区中有$k_1, k_2, \ldots, k_m$测试用例。初始化参数ε并且$\varepsilon > 0$。首先，根据每一个分区对应的概率p_i随机选取一个分区c_i（其中$p_1 + p_2 + \cdots + p_m = 1$），然后，在$c_i$中随机选取一个测试用例 t 并执行。如果 t 揭示了软件故障，增大c_i的选取概率p_i，减小其它分区的选取概率p_j（j ≠ i），移除检测到的故障。

$$p_j' = \begin{cases} p_j - \frac{\varepsilon}{m-1} & if\, p_j \geq \frac{\varepsilon}{m-1} \\ 0 & if\, p_j < \frac{\varepsilon}{m-1} \end{cases} \tag{2}$$

并且

$$p_i' = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{m} p_j' \tag{3}$$

如果 t 没有揭示软件故障，减小c_i的选取概率p_i，增大其它分区的选取概率p_j

（j ≠ i），移除检测到的故障。

$$p_i' = \begin{cases} p_i - \varepsilon & if\, p_i \geq \varepsilon \\ 0 & if\, p_i < \varepsilon \end{cases} \tag{4}$$

对任意$j = 1, 2, \ldots, m$并且$j \neq i$，有：

$$p_j' = \begin{cases} p_j + \frac{\varepsilon}{m-1} & if\, p_i \geq \varepsilon \\ p_j + \frac{p_i}{m-1} & if\, p_i < \varepsilon \end{cases} \tag{5}$$

DRT 的详细信息展示在图 2-2 中。DRT 首先根据测试剖面$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$随机选取一个分区$c_i$（算法 1 中的第 2 行），然后在$c_i$中随机选取一个测试用例 t（算法 1 中的第 3 行）。如果 t 揭示了软件中的故障，增大c_i的选取概率，减小其它分区c_j（$j = 1, 2, \ldots, m, j \neq i$）的选取概率（算法 1 中的第 6 行）。如果 t 没有揭示软件中的故障，减小c_i的选取概率，增大其它分区c_j（$j = 1, 2, \ldots, m, j \neq i$）的选取概率（算法 1 中的第 8 行）。由此可以看出 DRT 在测试的过程更新测试剖面，使得失效率大的分区选取的概率增大，失效率小的分区选取的概率减小。

Algorithm 1 DRT

Input: $\varepsilon, p_1, p_2, \ldots, p_m$

1. **while** termination condition is not satisfied
2. Select a partition c_i according to the test profile $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$.
3. Select a test case *t* form c_i.
4. Test the software using *t*.
5. **if** a fault is revealed by *t*
6. Update$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$according to Formulas 2 and 3.
7. **else**
8. Update$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$according to Formulas 4 and 5.
9. **end if**
10. **end while**

图 2-2 动态随机测试技术

动态随机测试的性能受两方面的影响：初始测试剖面和动态随机测试中的参数值。这方面代表性工作包括：（a）初始测试剖面：Li 等人提出了在测试过程中出现预定标准时测试剖面转换成理论最优剖面的最优动态随机测试技术[46]。（b）参数值：Yang 等人提出了在测试的过程中动态调整参数的方法[47]; Lv 等人通过理论分析的方式分析两个参数的比值在一定的区间时可以保证失效率大的分区选取概率不断增大[48]。

课题组将 DRT 应用到 Web 服务的测试中，提出了面向 Web 服务的动态随机测试框架，通过经验研究的方式验证了该技术的有效性[49,50]。此外，课题组在 DRT 的基础上，增加了参数的适应性机制，提出了适应性分区测试技术，并且分别利用 Markov 理论和奖惩机制提出了两种测试技术[7]：（a）基于

Markov 理论的适应性分区测试技术（MAPT），将分区视为状态，根据测试用例执行的结果更新状态转移概率；（b）基于奖惩机制的适应性分区测试技术（PAMT），旨在加速找到高失效率的分区，使得具有故障检测能力的测试用例更有可能被选中。接下来，详细介绍这两种测试技术。

（1） 基于 Markov 理论的适应性分区测试技术（MAPT）

Markov 随机过程称为 Markov 链，具备“无后效应”，即确定过程将来的状态只需知道目前的情况，并不需要对以往状况的认识。对任意$\mathrm{n} \geq 0$及状态$i, j, i_0, \ldots, i_{n-1}$，有

$$P\{X_{n+1}|X_0 = i_0, X_1 = i_1, \ldots, X_n = i_n\} = P\{X_{n+1} = \mathrm{j}|X_n = \mathrm{i}\}$$

由上式可得，一旦 Markov 链的初始分布$P\{X_{n+1} = i_0\}$给定，其统计特性完全由条件概率$P\{X_n = i_n|X_{n-1} = i_{n-1}\}$决定。

假设状态空间$S = \{1, 2, \ldots, m\}$。

定义 6（转移概率） 条件概率$P\{X_n = \mathrm{j}|X_{n-1} = i\}$为 Markov 链的一部转移概率，简称转移概率。

定义 7（时齐 Markov 链） 当 Markov 链的转移概率只与状态i, j有关，与n无关时，称 Markov 链为时齐的，记为$p_{ij} = P\{X_{n+1} = \mathrm{j}|X_n = \mathrm{i}\}$。

由定义 7，可以将p_{ij}排成一个矩阵的形式：

$$P = p_{ij} = \begin{pmatrix} p_{11} & \cdots & p_{1m} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mm} \end{pmatrix}$$

则称P为转移矩阵。

MAPT 在动态随机测试的基础上增加了转移概率的思想：将每一次选中的分区当作目前系统所处的状态，根据该分区测试用例的执行结果调整到下一个状态（分区）的概率。如果当前选中分区的测试用例揭示了软件的故障，那么就增大下一步处于该状态的概率，减小到其它状态的概率。如果没有揭示软件中的故障，减小下一步处于该状态的概率，增大到其它状态的概率。另一方面，每一个分区每次增大或者减少的幅度应当不同。如果一个分区选取的概率较大，说明在以往的测试过程中揭示的故障数目较多。相反如果一个分区选取的概率较小，说明在以往测试过程中揭示的故障数目较少甚至没有揭示故障。因此增大或者减小分区的选取概率应考虑当前概率的大小。整个测试过程可以用转移矩阵表示。

假设将软件的输入域划分为m不相交的分区$c_1, c_2, \ldots, c_m$，每一个分区中有$k_1, k_2, \ldots, k_m (k_1 + k_2 + \cdots + k_m = k)$测试用例。初始化参数$\beta$并且$\beta > 0$。在测试的过程中，如果将每个时间点$time(time = 0, 1, 2, \ldots)$测试用例所在的分

区作为时刻$time$所处的状态，则整个状态空间$S=\{s_{time}|time\geq 0\}=\{c_1,c_2,\ldots,c_m\}$，将每一个时间点测试用例的执行状况按照某种测试技术调整对应分区的转移概率作为一次决策行动，那么各时刻的行动全体构成行动空间$\mathrm{A}=\{a_{time}|time\geq 0\}=\{1,2,\ldots,k\}$，每一个时间点的状态$s_{time}$和采取的行动$a_{time}$会影响到下一个时间点的状态$s_{time+1}$。不妨设初始时刻的状态转移矩阵$p_{ij}$如下：

$$p_{ij}=\begin{pmatrix}\frac{1}{m} & \cdots & \frac{1}{m}\\ \vdots & \ddots & \vdots\\ \frac{1}{m} & \cdots & \frac{1}{m}\end{pmatrix}$$

当第一次执行时根据测试剖面$\{<c_1,p_1>,<c_2,p_2>,\ldots,<c_m,p_m>\}$，随机选取一个分区$c_i$，然后根据测试结果调整第 i 行的概率，并根据状态 i 的转移概率选取下一个状态。具体的调整规则如下：如果c_i分区的测试用例 t 揭示了软件故障，增大下一步处于c_i状态的概率，减小转移到其它状态的概率，并移除故障：

$$p'_{ij}=\begin{cases}p_{ij}-\frac{\beta\times p_{ii}}{m-1} & if\,p_{ij}\geq\frac{\beta\times p_{ii}}{m-1}\\ 0 & if\,p_j<\frac{\beta\times p_{ii}}{m-1}\end{cases} \tag{6}$$

并且

$$p'_{ii}=1-\sum_{\substack{j=1\\ j\neq i}}^{m}p'_{ij} \tag{7}$$

如果c_i分区的测试用例 t 没有揭示软件故障，减小下一步处于c_i状态的概率，增大转移到其它状态的概率，并移除故障：

$$p'_{ij}=\begin{cases}p_{ij}-\frac{\beta\times p_{ij}}{m-1} & if\,p_{ii}\geq\frac{\beta\times(1-p_{ii})}{m-1}\\ p_{ij} & if\,p_{ii}<\frac{\beta\times(1-p_{ii})}{m-1}\end{cases} \tag{8}$$

对任意$j=1,2,\ldots,m$并且$j\neq i$，有：

$$p'_{ii}=\begin{cases}p_{ii}-\frac{\beta\times(1-p_{ii})}{m-1} & if\,p_{ii}\geq\frac{\beta\times(1-p_{ii})}{m-1}\\ p_{ii} & if\,p_{ii}<\frac{\beta\times(1-p_{ii})}{m-1}\end{cases} \tag{9}$$

MAPT 的详细信息展示在图 2-3 中。MAPT 首先根据测试剖面$\{<c_1,p_1>,<c_2,p_2>,\ldots,<c_m,p_m>\}$随机选取一个分区$c_i$（算法 2 中的第 5 行），然后在$c_i$中随机选取一个测试用例 t（算法 2 中的第 6 行）。如果 t 揭示了软件中的故障，增大c_i的选取概率，减小其它分区c_j（$\mathrm{j}=1,2,\ldots,\mathrm{m},\mathrm{j}\neq\mathrm{i}$）的选取概率（算法 2 中的第 13 行）。如果 t 没有揭示软件中的故障，减小c_i的选取概率，增大其它分区c_j（$\mathrm{j}=1,2,\ldots,\mathrm{m},\mathrm{j}\neq\mathrm{i}$）的选取概率（算法 2 中的第 13 行）。

（2） 基于奖惩机制的适应性分区测试技术（PAPT）

基于奖惩机制的适应性分区测试技术（RAPT）旨在加速找到具有较高故障检测能力的分区。该技术的主要思想是：如果某一个分区的测试用例揭示了软件的故障，认为该分区的其它测试用例有较高的概率再次揭示故障。因此下一次仍在该分区中选择测试用例，并与该分区绑定的奖励因子增加 1，

Algorithm 2 MAPT

Input: $\beta, p_1, p_2, \dots, p_m$

1. Initialize Markov matrix P by setting $p_{ij} = p_j$.
2. Set $noTC = 0$.
3. **while** termination condition is not satisfied
4. **if** $noTC = 0$
5. Select a partition c_i according to the test profile $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$.
6. Select a test case *t* form c_i.
7. **else**
8. Given that the previous test case is from c_i, select a partition c_j according to Profile $\{p_{i1}, p_{i2}, \dots, p_{im}\}$.
9. Select a test case t from c_j.
10. **end if**
11. Test the software using *t*.
12. Increment $noTC$ by 1.
13. Update p based on the testing result according to Formulas 6 to 9.
14. **end while**

图 2-3 基于 Markov 理论的适应性分区测试技术

对应的惩罚因子清 0，重复在该分区中选择测试用例直到没有揭示故障；如果一个分区的惩罚因子累计到一定数目，认为该分区的试效率很低甚至为 0，设置该分区的选取概率为 0。该技术的具体算法如下：

假设将软件的输入域划分为m不相交的分区$c_1, c_2, \dots, c_m$，每一个分区中有$k_1, k_2, \dots, k_m (k_1 + k_2 + \cdots + k_m = k)$测试用例。初始化参数$\beta$并且$\beta > 0$，并且每个分区的奖励因子$reward_h = 0$，惩罚因子$punishment_h = 0$，其中$h \in \{1, 2, \dots, m\}$。根据每一个分区对应的概率$p_i$随机选取一个分区$c_i$（其中$p_1 + p_2 + \cdots + p_m = 1$），然后，在$c_i$中随机选取一个测试用例 t 并执行。如果 t 揭示了软件故障，$reward_i$增加 1，惩罚因子$punishment_i$清 0，下一个测试用例仍在该分区选择，重复以上步骤直到没有揭示故障然后更新测试剖面。如果 t 没有揭示软件故障，$reward_i$清 0，惩罚因子pu_i加 1 并且惩罚因子到达阈值Bou时$p_i = 0$。

如果$reward_i > 0$，则：

$$p_j' = \begin{cases} p_j - \frac{\beta\times(1+\ln reward_i)}{m-1} & if p_j \geq \frac{\beta\times(1+\ln reward_i)}{m-1} \\ 0 & if p_j < \frac{\beta\times(1+\ln reward_i)}{m-1} \end{cases} \quad (10)$$

并且

$$p_i' = 1 - \sum_{\substack{j=1 \\ j\neq i}}^{m} p_j' \quad (11)$$

如果$reward_i = 0$，则：

$$p_i' = \begin{cases} p_i - \beta & if p_i \geq \beta \\ 0 & if p_i < \beta \ \ or \ \ pu_i = Bou \end{cases} \quad (12)$$

对任意$j = 1, 2, \ldots, m$并且$j \neq i$，有

$$p_j' = \begin{cases} p_j + \frac{\beta}{m-1} & if p_i \geq \beta \\ p_j + \frac{p_i}{m-1} & if p_i < \beta \ \ or \ \ pu_i = Bou \end{cases} \quad (13)$$

PAPT 的详细信息展示在图 2-4 中。PAPT 首先根据测试剖面$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$随机选取一个分区$c_i$（算法 3 中的第 4 行），然后在$c_i$中随机选取一个测试用例 t（算法 3 中的第 5 行）。如果 t 揭示了软件中的故障，增大c_i对应的奖励因子$reward_i$（算法 3 中的第 8 行），并将对应的惩罚因子pu_i的值设置为 0（算法 3 中的第 9 行），持续在c_i中选择测试用例直到该测试用例没有揭示软件中的故障。如果$reward_i$的值不为 0，增大c_i的选取概率，减小其它分区的选取概率（算法 3 的第 15 行）。如果$reward_i$的值为 0，减小c_i的选取概率，增大其它分区的选取概率（算法 3 的第 18 行）。

2.2.3 课题组前期研究工作

课题组前期研究工作在面向 Web 服务的蜕变测试与蜕变关系获取方面，取得了如下成果：

（1） 蜕变测试与其它技术结合

Sun 等人利用蜕变测试技术成功地测试了缺少测试预期的 Web 服务，提出了面向 Web 服务的蜕变测试框架[34,51]，依据该框架开发了面向 Web 服务的蜕变测试支持框架 MT4WS，支持定义蜕变关系、生成测试用例、执行测试用例、验证测试结果、配置测试信息和生成测试报告等功能。Sun 等人改进了迭代蜕变测试技术[52]，提出了面向 Web 服务的迭代蜕变测试技术[53]，首先原始测试用例集（依据其它测试用例生成技术获得）依据蜕变关系产生衍生测试用例，判断生成的衍生测试用例的输入是否符合输入规范，若符合输入规范将衍生测试用例添加到原始测试用例集中；若不符合输入规范，舍弃该衍生测试用例。按照上述步骤循环一定的次数（测试者预先定义循环的次数）得到最终的原始测试用例集。该技术在时间和规模方面提高了 Web 服务测试

用例的生成效率，采用经验研究的方式验证了该技术生成的测试用例的故障检测效率优于随机生成的测试用例。

（2） 蜕变关系获取方法

在蜕变关系获取方面，Sun 等人提出了一种基于数据变异的蜕变关系获取技术[54]，结合蜕变测试的进本原理与数据变异的特点，总结了 7 类数值型变异算子以及相应的蜕变关系映射规则，提高了蜕变关系的获取效率，开发了相应的支持工具并集成到 MT4WS 工具中，提高了蜕变测试的自动化程度。

Algorithm 3 PAPT

Input: $\beta, p_1, p_2, \ldots, p_m$

1. Initialize $reward_i = 0$ and $pu_i = 0$ for all i = 1,2, … , m.
2. Set $noTC = 0$.
3. **while** termination condition is not satisfied
4. Select a partition c_i according to the test profile $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$.
5. Select a test case *t* form c_i.
6. Test the software using *t*.
7. **while** a fault is revealed by *t*
8. Increment $reward_i$ by 1.
9. Set $pu_i = 0$.
10. Select a test case *t* form c_i.
11. Test the software using *t*.
12. **end while**
13. Increment pu_i by 1.
14. **if** $reward_i \neq 0$
15. Update$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$according to Formulas 10 and 11, respectively.
16. Set $reward_i = 0$.
17. **else**
18. Update$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$according to Formulas 12 and 13, respectively.
19. Set $reward_i = 0$.
20. **end if**
21. **end while**

图 2-4 基于奖惩机制的适应性分区测试技术

2.3 小结

本章首先介绍了蜕变测试、随机测试、分区测试和变异测试的相关概念，然后介绍了蜕变测试和适应性分区测试技术的国内外研究现状，最后介绍了课题组在蜕变测试以及适应性分区测试领域的研究工作。

3 蜕变测试与适应性分区测试的集成方法

本章首先分析蜕变测试技术与适应性分区测试技术集成的必要性，然后介绍提出的两种适应性蜕变测试技术，包括方法框架、算法。

3.1 蜕变测试与适应性分区测试集成的必要性

传统的蜕变测试技术随机地执行测试用例，该方式简单、易于实现，但是没有有效地利用测试过程信息。适应性分区测试引入了反馈机制，根据当前测试用例的执行结果更新测试剖面，提升了随机执行测试用例的故障检测效率。既然适应性分区测试的故障检测效率高于随机地执行测试用例，本文探索蜕变测试技术与适应性分区测试的集成方法，提出了以蜕变关系为中心的适应性蜕变测试技术（M-AMT）。

适应性分区测试假设存在测试预期。实际情况中，测试预期可能不存在或者使用测试预期需要极高的代价。在缺少测试预期时，传统的适应性分区测试不能用来测试软件，即适应性分区测试的实用性受到了限制。为此，本文将蜕变关系用于适应性分区测试中测试结果的判定机制，提出了以分区为中心的适应性蜕变测试技术（P-AMT）。

分别介绍以蜕变关系为中心的适应性蜕变测试技术和以分区为中心的适应性蜕变测试技术。

3.2 以蜕变关系为中心的适应性蜕变测试技术（M-AMT）

本节首先介绍 M-AMT 的框架，然后介绍一种更新测试剖面的策略（MDRT），最后描述 M-AMT 的测试过程。

3.2.1 以蜕变关系为中心的适应性蜕变测试框架

蜕变关系是蜕变测试技术的一个明显特征，测试人员通过原始测试用例与衍生测试用例的执行结果是否满足蜕变关系判断待测程序是否存在故障。对于复杂的程序可能存在若干条蜕变关系，通过尽早地执行具有揭示故障能力的蜕变关系相关的原始测试用例与衍生测试用例可以有效地节约测试资源并提高蜕变测试的测试效率。

适应性分区测试在测试的过程中动态改变测试剖面，不断增大失效率大的分区对应的选取概率，同时减小失效率小的分区对应的选取概率。本文利

用适应性分区测试技术的上述特点，通过选择失效率大的分区，指导性地选择下一个蜕变关系。图 3-1 展示了以蜕变关系为中心的适应性蜕变测试框架，主要包含六个部分：创建完整测试帧、构建分区、选择蜕变关系、生成与执行测试用例、更新测试剖面、选择分区。

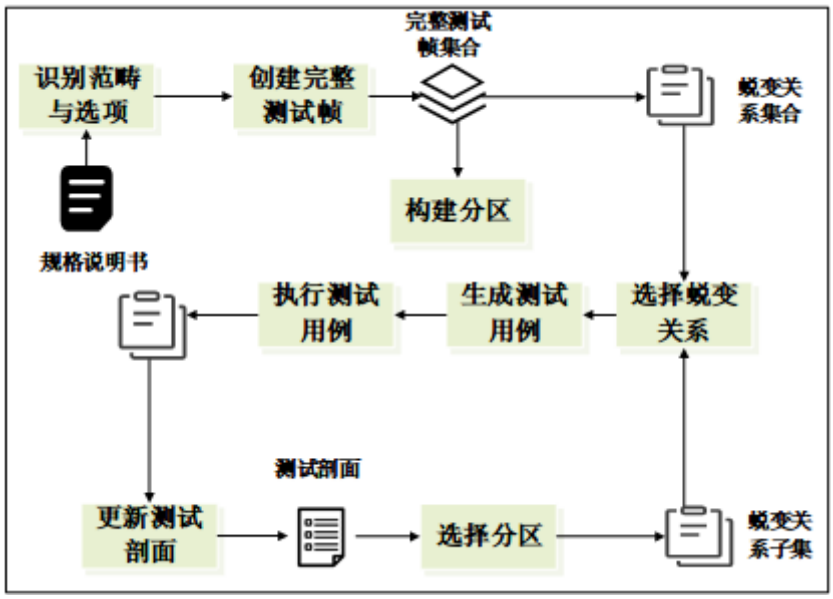


图 3-1 以蜕变关系为中心的适应性蜕变测试框架

（1） 创建完整测试帧：从规格说明书中，识别范畴和选项，有效地组合不同范畴中的选项，构建完整测试帧。

（2） 构建分区：实际上，每一个完整测试帧代表一系列的输入模式，因此每一个测试帧可以作为一个分区。另一方面，测试人员可以根据具体的情况或者自身的测试经验，考虑最重要范畴并将不同范畴中的选项有效地组合，得到粗粒度的分区。

（3） 选择蜕变关系：第一次执行测试时，测试人员可以在蜕变关系集中随机地选择一个蜕变关系，然后根据执行的结果调整测试剖面， 依据调整后的测试剖面选择分区，从选取分区中所有测试帧涉及到的蜕变关系集中随机选取一个蜕变关系。

（4） 生成与执行测试用例：根据选择的蜕变关系找到相应的两个完整测试帧，实例化两个完整测试帧，得到两个测试用例（原始测试 用例和衍生测试用例），然后执行测试用例。

（5） 更新测试剖面：判断测试用例的执行结果是否违反了蜕变关系，然后判断原始测试用例和衍生测试用例是否属于同一分区，根据判断的结果选择更新测试剖面的技术。

（6） 选择分区：依据测试剖面选择分区，然后将选取分区所有测试帧涉及到的蜕变关系组成候选蜕变关系集。

本文重点解决 M-AMT 的如下问题：

（1） 更新测试剖面：如果当前执行的原始测试用例与衍生测试用例属于同一个分区，传统的动态随机测试可以用来更新测试剖面。传统的动态随机测试增大/减小一个分区的选取概率，减小/增大其它分区的选取概率。当原始测试用例与衍生测试用例不属于同一分区时，依据测试结果需要增大/减小二个分区的选取概率，减小/增大其它分区的选取概率。在此情况下，传统的动态随机测试不再能调整测试剖面。

（2） 选择蜕变关系：以蜕变关系为中心的适应性蜕变测试技术利用适应性分区测试技术具有快速找到较高失效率分区的特点，旨在 尽快选中 具有揭示故障能力的蜕变关系，提高蜕变测试的测试效率。如何在测试的过程中选择蜕变关系，影响蜕变测试的测试效率。

下面，讨论测试剖面更新问题，解决问题（1）；然后介绍 M-AMT 的算法，展示了蜕变关系的选择过程，解决问题（2）。

3.2.2 测试剖面更新策略（MDRT）

传统的动态随机测试技术每次执行一个测试用例tc，根据tc的执行结果调整tc所在分区p_i的选取概率以及其它分区的概率。在蜕变测试技术中，每次执行测试涉及两个测试用例：原始测试用例stc（采用测试用例生成技术得到）和衍生测试用例ftc（根据蜕变关系与原始测试用例得到）。当stc与ftc属于不同的分区时，本文采用如下测试剖面更新策略。

假设将软件的输入域划分为m不相交的分区$\{c_1, c_2, \dots, c_m\}$，每一个分区有$k_1, k_2, \dots, k_m$测试帧，其中$m > 2$。在 METRIC 方法中，测试人员通过对比两个不同的、有效的完整测试帧tf_g^i和tf_h^n，识别蜕变关系MR_q，然后将tf_g^i和tf_h^n实例化得到原始测试用例stc_g^i和衍生测试用例ftc_h^n，其中tf_g^i表示c_i分区第 g 个测试帧，stc_g^i表示将c_i分区第 g 个测试帧实例化后得到的测试用例。当$i \neq n$时，根据stc_g^i和ftc_h^n的测试结果是否违反蜕变关系MR_q，调整测试剖面$\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$的方式如下：

- 如果stc_g^i和ftc_h^n的测试结果违反了蜕变关系MR_q，每一个分区的选取概率调整如下：

$$p_j' = \begin{cases} p_j - \frac{2\varepsilon}{m-2} & if p_j \geq \frac{2\varepsilon}{m-2} \ and \ j \neq i, j \neq n \\ 0 & if p_j < \frac{2\varepsilon}{m-2} and \ j \neq i, j \neq n \end{cases} \quad (14)$$

并且

$$p_i' = p_i + \frac{\left(1 - \sum_{j=1, j\neq i, j\neq m}^{m} p_j'\right) - p_i - p_n}{2} \quad (15)$$

$$p_n' = p_n + \frac{\left(1 - \sum_{j=1, j\neq i, j\neq m}^{m} p_j'\right) - p_i - p_n}{2} \quad (16)$$

- 如果stc_g^i和ftc_h^n的测试结果没有违反蜕变关系MR_q，则每一个分区的选取概率调整如下：

$$p_i' = \begin{cases} p_i - \varepsilon & if p_i \geq \varepsilon \\ 0 & if p_i < \varepsilon \end{cases} \quad (17)$$

$$p_n' = \begin{cases} p_n - \varepsilon & if p_i \geq \varepsilon \\ 0 & if p_i < \varepsilon \end{cases} \quad (18)$$

对任意$j = 1,2,\dots,m$并且$j \neq i, j \neq m$，有：

$$p_j' = p_j + \frac{(p_i - p_i') + (p_n - p_n')}{m-2} \quad (19)$$

MDRT 根据原始测试用例与衍生测试用例的执行结果是否满足蜕变关系来更新测试剖面，具体信息如图 3-2。不妨假设原始测试用例stc_g^i与衍生测试用例ftc_h^n分别来自分区c_i和c_n并且$i \neq n$。如果原始测试用例stc_g^i与衍生测试用例ftc_h^n的测试结果违反了蜕变关系（算法 4 中第 2 行），增大c_i和c_n的选取概率p_i和p_n，减小其它分区的选取概率$p_j (j = 1,2,\dots,m \ and \ j \neq i, j \neq n)$（算法 4 中第 2 行）。如果原始测试用例$stc_g^i$与衍生测试用例$ftc_h^n$的测试结果没有违反蜕变关系（算法 4 中第 3 行），减小c_i和c_n的选取概率p_i和p_n，增大其它分区的选取概率$p_j (j = 1,2,\dots,m \ and \ j \neq i, j \neq n)$（算法 4 中第 4 行）。

Algorithm 4 MDRT

Input: $\varepsilon, p_1, p_2, \dots, p_m$

1. **if** Results of $stc_g^i \ and \ ftc_h^n$ violate the metamorphic relation MR_q
2. Update$\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 14, 15, and 16.
3. **else**
4. Update$\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 17,18, and 19.
5. **end if**

图 3-2 测试剖面更新策略

3.2.3 M-AMT 算法

以蜕变关系为中心的适应性蜕变测试技术在测试的过程中根据当前测试

信息选择下一个蜕变关系，利用适应性分区测试可以高效揭示故障的特点，控制测试的执行过程，提高蜕变测试的测试效率。

假设待测软件的输入域划分为m不相交的分区$c_1, c_2, \dots, c_m$，每一个分区中有$f_1, f_2, \dots, f_m$测试帧并且每一个测试帧实例化一个测试用例，分区$c_i(i = 1,2, \dots, m)$中所有测试帧涉及到的蜕变关系组成候选蜕变关系集$MR_i(i = 1,2, \dots, m)$，所有的蜕变关系组成集合$\mathrm{MRs} = MR_1 \cup MR_2 \cup \cdots \cup MR_m$。第一个蜕变关系$MR'$是在MRs中随机地选择，并实例化$MR'$涉及到的两个不同、有效的完整测试帧$tf_g^l$和$tf_h^n$，得到原始测试用例$tc_g^l$和衍生测试用例$tc_h^n$，其中$tf_g^l$表示分区$c_g(g = 1,2,3, \dots, \mathrm{m})$中的第$l$测试帧，$tc_g^l$表示测试帧$tf_g^l$实例化得到的测试用例。然后执行$tc_g^l$和$tc_h^n$，依据执行的结果是否违反蜕变关系更新测试剖面：

- 如果tc_g^l和tc_h^n的执行结果违反了蜕变关系，
  (1) 当tc_g^l和tc_h^n属于不同的分区时，依据 MDRT 中的公式 13、14 和 15 更新测试剖面。
  (2) 当tc_g^l和tc_h^n属于同一分区时，依据 DRT 中的公式 1 和 2 更新测试剖面。
- 如果tc_g^l和tc_h^n的执行结果没有违反蜕变关系，
  (1) 当tc_g^l和tc_h^n属于不同的分区时，依据 MDRT 中的公式 16、17 和 18 更新测试剖面。
  (2) 当tc_g^l和tc_h^n属于同一分区时，依据 DRT 中的公式 3 和 4 更新测试剖面。

执行完第一个测试用例之后，判断是否满足终止条件。如果不满足终止条件，依据测试剖面选择分区c_i，然后从c_i对应的蜕变关系集MR_i中随机选取一个蜕变关系MR'，实例化MR'涉及到的两个不同、有效的完整测试帧tf_g^l和tf_h^n并得到原始测试用例tc_g^l和衍生测试用例tc_h^n，然后执行tc_g^l和tc_h^n，依据执行的结果是否违反蜕变关系更新测试剖面：

- 如果tc_g^l和tc_h^n的执行结果违反了蜕变关系，
  (1) 当tc_g^l和tc_h^n属于不同的分区时，依据 MDRT 中的公式 14、15 和 16 更新测试剖面。
  (2) 当tc_g^l和tc_h^n属于同一分区时，依据 DRT 中的公式 2 和 3 更新测试剖面。
- 如果tc_g^l和tc_h^n的执行结果没有违反蜕变关系，
  (1) 当tc_g^l和tc_h^n属于不同的分区时，依据 MDRT 中的公式 17、18 和

19 更新测试剖面。

(2) 当tc_g^l和tc_h^n属于同一分区时，依据 DRT 中的公式 4 和 5 更新测试剖面。

M-AMT 选择蜕变关系与执行测试用例的详细过程展示在算法 5 中，如图 3-3。首先，第一个蜕变关系MR'是在MRs中随机地选择，然后，将 MR'涉及的两个测试帧实例化得到原始测用例和衍生测试用例，执行测试用例。如果执行结果违反了蜕变关系，根据两个测试用例是否属于同一个分区更新测试剖面（算法 5 的 6、7、8、9、10 和 11 行）。如果测试用例的执行结果没有违反蜕变关系，根据两个测试用例是否属于同一个分区更新测试剖面（算法 5 的 12、13、14、15、16 和 17 行）。测试人员判断测试系统的状态是否满足终止条件，如果满足则终止测试；如果不满足终止条件，依据测试剖面$\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$，选取分区$c_i$，从分区$c_i$对应的候选蜕变关系集中选择蜕变关系$MR'$（算法 5 中 20 行），然后，实例化$MR'$涉及的两个测试帧，并执行得到的两个测试用例（算法 5 中 21 行）。如果测试用例的执行结果违反了蜕变关系，根据两个测试用例是否属于同一个分区更新测试剖面（算法 5 的 22、23、24、25、26 和 27 行）。如果测试用例的执行结果没有违反蜕变关系，根据两个测试用例是否属于同一个分区更新测试剖面（算法 5 的 28、29、30、31、32 和 33 行）。上述算法中的终止条件（termination condition）是测试结束满足的条件。本文设置了两种终止条件：（a）检测出第一个故障；（b）执行完给定数目的测试用例。

3.3 以分区为中心的适应性蜕变测试技术（P-AMT）

本节介绍以分区为中心的适应性蜕变测试框架与过程。

3.3.1 以分区为中心的适应性蜕变测试框架

适应性分区测试技术利用当前测试信息动态更新测试剖面，增大失效率大的分区的选取概率，减小失效率低的分区的选取概率，使得失效率大的分区更有可能被选中，提高了测试效率。传统的适应性分区测试假设测试预期存在。但是，实际中很多程序的测试预期不存在或者运用测试预期需要很大的代价。当得到或运用测试预期较难时，适应性分区测试技术不再能对程序进行测试。从而限制了适应性分区测试技术的应用。为了解决缺少测试预期时适应性分区测试不能测试程序的问题，本文探索了适应性分区测试与蜕变测试的集成方法。

Algorithm 5 M-AMT

Input: $\varepsilon, p_1, p_2, \ldots, p_m, MR_1, MR_2, \ldots, MR_m, MRs, counter$

1. Initialize $counter$ = 1.
2. **while** termination condition is not satisfied
3. **if** $counter$ == 1
4. Select MR' from MRs and increment counter by 1.
5. Transform test frames tf_g^l and tf_h^n into test cases tc_g^l and tc_h^n, and execute test cases tc_g^l and tc_h^n.
6. **if** results of tc_g^l and tc_h^n violate MR'
7. **if** tc_g^l and tc_h^n belong to different partitions
8. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 14, 15, and 16.
9. **else**
10. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 2 and 3.
11. **end if**
12. **else**
13. **if** tc_g^l and tc_h^n belong to same partition
14. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 17, 18, and 19.
15. **else**
16. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 4 and 5.
17. **end if**
18. **end if**
19. **else**
20. Select a partition c_i according to test profile $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$, and then select a MR' form MR_i.
21. Transform test frames tf_g^l and tf_h^n into test cases tc_g^l and tc_h^n, and execute test cases tc_g^l and tc_h^n.
22. **if** results of tc_g^l and tc_h^n violate MR'
23. **if** tc_g^l and tc_h^n belong to different partitions
24. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 14, 15, and 16.
25. **else**
26. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 2 and 3.
27. **end if**
28. **else**
29. **if** tc_g^l and tc_h^n belong to same partition
30. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 17, 18, and 19.
31. **else**
32. Update $\{< c_1, p_1 >, < c_2, p_2 >, \ldots, < c_m, p_m >\}$ according to Formulas 4 and 5.
33. **end if**
34. **end if**
35. **end if**
36. **end while**

图 3-3 M-AMT 算法

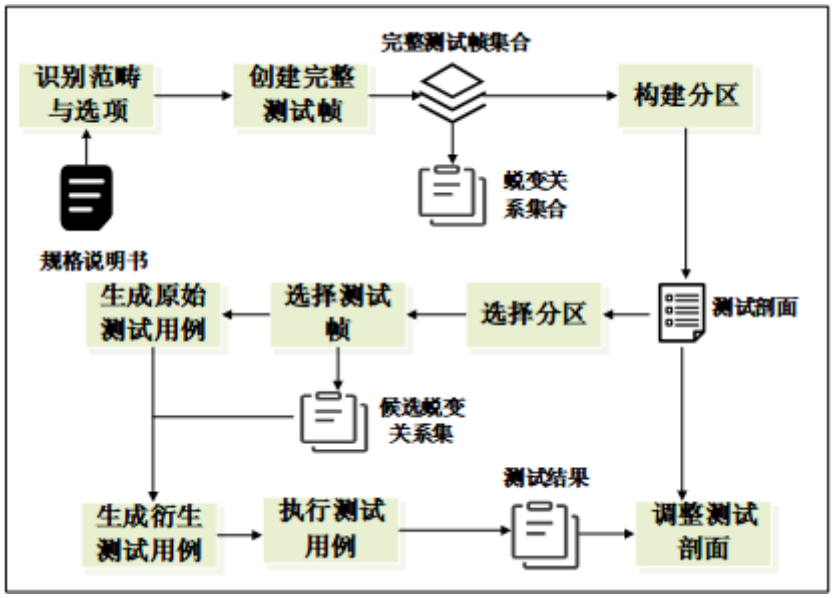


图 3-4 以分区为中心的适应性蜕变测试框架

蜕变测试利用待测程序存在的某种内在属性（蜕变属性）生成测试用例（包括原始测试用例和衍生测试用例），通过原始测试用例与衍生测试用例对应的输出是否满足某种关系（蜕变关系）进行测试结果判定。蜕变测试无须知道某个测试用例的预期输出，因此可以有效缓解测试预期问题。本文利用蜕变测试无需测试预期的特点，验证测试用例的输出是否违反蜕变关系，并根据验证的结果更新测试剖面。图 3-4 展示了以分区为中心的适应性蜕变测试技术方法框架，包含如下六个部分：

（1） 创建完整测试帧：从规格说明书中，识别范畴和选项，有效地组合不同范畴中的选项，构建完整测试帧。

（2） 构建分区：实际上，每一个完整测试帧代表一系列的输入模式，因此每一个测试帧可以作为一个分区。另一方面，测试人员可以根据具体的情况或者自身的测试经验，考虑最重要范畴中的选项并将这些 选项有效地组合，得到粗粒度的分区。

（3） 选择分区：依据测试剖面选择分区，并且在测试的过程中根据不同分区测试用例揭示故障的情况调整分区选取的概率，揭示故 障数目多的分区的选取概率变大，揭示故障数目少的分区的选取概率减小。

（4） 生成和选择测试用例：在选中的分区中随机地选择一个测试帧，将选中的测试帧实例化得到测试用例，依据相关的蜕变关系得到衍生测试用例。

（5） 执行测试用来并验证结果：使用上一步获得的测试用例（包括原始测试用例与衍生测试用例）执行待测程序，然后验证执行的 结果是否违反蜕

变关系。

（6） 更新测试剖面：根据原始测试用例与衍生测试用例的执行结果是否违反蜕变关系更新测试剖面，增大失效率高的分区的选取概 率，减小失效率低的分区的选取概率。

本文主要解决以分区为中心的适应性蜕变测试的如下问题：

（1） 生成和选择测试用例：依据测试剖面选择分区之后，选择原始测试用例以及与之相关的蜕变关系是一个待解决的问题。

（2） 更新测试剖面：如果执行的原始测试用例与衍生测试用例属于同一个分区，传统的动态随机测试可以用来更新测试剖面。传统的动态随机测试增大/减小一个分区的选取概率，减小/增大其它分区的选取概率。当原始测试用例与衍生测试用例不属于同一分区时，依据测试结果需要增大/减小二个分区的选取概率，减小/增大其它分区的选取概率。在此情况下，传统的动态随机测试不再能调整测试剖面。

3.3.2 P-AMT 算法

P-AMT 根据当前执行的测试用例（包含原始测试用例与衍生测试用例）是否违反蜕变关系更新测试剖面，最主要的特点：将测试用例的执行结果是否违反蜕变关系作为一种判断待测程序是否存在故障的机制，并尽可能多地选取失效率大的分区来提高故障检测效率。

假设待测软件的输入域划分为m不相交的分区$c_1, c_2, \dots, c_m$，每一个分区中有$f_1, f_2, \dots, f_m$测试帧（每一个测试帧实例化一个测试用例）和$k_1, k_2, \dots, k_m$个测试用例并且$k_1 + k_2 + \cdots + k_m = k$，每一个测试用例$tc_i^s$(i = 1,2, … , m, and s > 0)涉及到的蜕变关系组成候选蜕变关系集MR_i^s，其中s为c_i分区第s测试用例，所有的蜕变关系组成集合 MRs。首先，依据测试剖面选择分区c_i，然后，在c_i中随机选取测试用例tc_i^s，将tc_i^s与候选蜕变关系集MR_i的蜕变关系依次作用得到候选测试用例对集合$\{< tc_i^s, ftc_i^1 >, < tc_i^s, ftc_i^2 >, \dots, < tc_i^s, ftc_i^{|MR_i^s|} >\}$，其中$|MR_i^s|$表示$MR_i^s$中蜕变关系的数目。分别执行每一个测试用例对并依据执行结果是否违反蜕变关系更新测试剖面：

- 如果tc_i^s和ftc_i^s的执行结果违反了蜕变关系，
  (1) 当tc_i^s和ftc_i^s属于不同的分区时，依据 MDRT 中的公式 14、15 和 16 更新测试剖面。
  (2) 当tc_i^s和ftc_i^s属于同一分区时，依据 DRT 中的公式 2 和 3 更新测试剖面。

- 如果tc_i^s和ftc_i^s的执行结果没有违反蜕变关系，
  (1) 当tc_i^s和ftc_i^s属于不同的分区时，依据 MDRT 中的公式 17、18 和 19 更新测试剖面。
  (2) 当tc_i^s和ftc_i^s属于同一分区时，依据 DRT 中的公式 4 和 5 更新测试剖面。

P-AMT 选择分区与执行测试用例的详细过程展示在算法 6 中，如图 3-5。首先，根据测试剖面选择分区c_i（算法 6 中第 2 行），然后，在c_i中随机挑选测试用例tc_i^s（算法 6 中第 3 行），依据与tc_i^s相关的蜕变关系MR'生成衍生测试用例ftc_i^s（算法 6 中第 5 行），执行tc_i^s和ftc_i^s（算法 6 中第 6 行）。如果测试用例的执行结果违反了蜕变关系，依据两个测试用例是否属于同一个分区更新测试剖面（算法 6 的 7、8、9、10、和 11 行）。如果测试用例的执行结果没有违反蜕变关系，依据两个测试用例是否属于同一个分区更新测试剖面（算法 6 的 13、14、15、16 和 17 行）。上述算法中的终止条件（termination condition）是测试结束满足的条件。本文设置了两种终止条件：（a）检测出第一个故障；（b）执行完给定数目的测试用例。

Algorithm 6 P-AMT

Input: $\varepsilon, p_1, p_2, \dots, p_m, MR_1, MR_2, \dots, MR_k, MRs$
1. **while** termination condition is not satisfied
2. Select a partition c_i according to test profile $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$.
3. Select a test case tc_i^s from c_i.
4. **for** all $MR' \in MR_i^s$
5. Generate ftc_i^s according to MR'.
6. Test the software using tc_i^s and ftc_i^s.
7. **if** results of tc_i^s and ftc_i^s violate MR'
8. **if** tc_i^s and ftc_i^s belong to different partitions
9. Update $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 13, 14, and 15.
10. **else**
11. Update $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 1 and 2.
12. **end if**
13. **else**
14. **if** tc_i^s and ftc_i^s belong to different partitions
15. Update $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 16, 17, and 18.
16. **else**
17. Update $\{< c_1, p_1 >, < c_2, p_2 >, \dots, < c_m, p_m >\}$ according to Formulas 3 and 4.
18. **end if**
19. **end if**
20. **end for**
21. **end while**

图 3-5 P-AMT 算法

3.4 小结

本章讨论了如何将蜕变测试与适应性分区测试集成的问题，提出了两种适应性蜕变测试技术，其中，M-AMT 侧重决解如何提高 MT 故障检测效率的问题；P-AMT 侧重解决 APT 在测试预期不存在情形下的应用问题。

4 适应性蜕变测试工具 APT2MT 设计与实现

本章介绍适应性蜕变测试技术支持工具 APT2MT 的设计与实现，包括需求分析、设计和实现，最后用一个实例进行演示。

4.1 需求分析

为了提高适应性蜕变测试技术的自动化程度，本文开发了相应的支持工具 APT2MT，支持以下四个方面：（a）配置适应性蜕变测试，包括设置 DRT 以及 MDRT 的参数、划分分区和选择测试技术；（b）输入终止条件；（c）查询变异体信息；（d）执行测试。基于 UML 用例图的 APT2MT 需求分析结果，如图 4-1 所示。

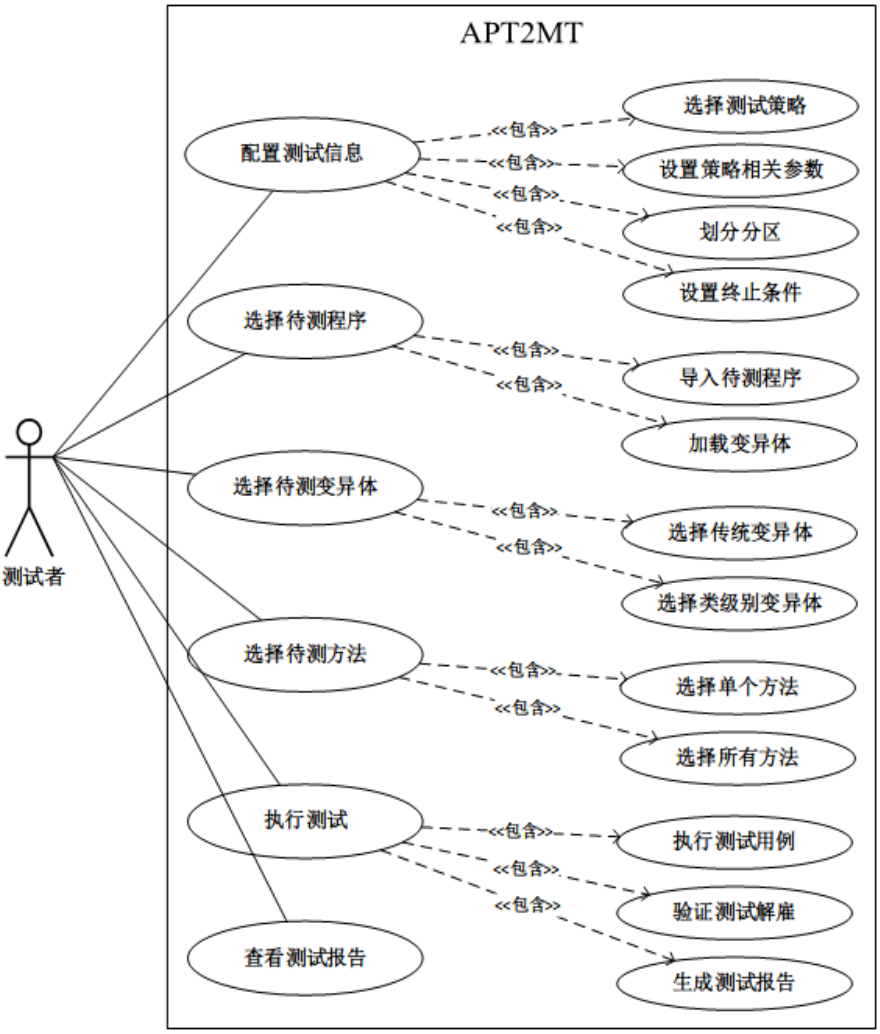


图 4-1 APT2MT 用例图

下面具体描述图 4-1 中的各个用例。

（1）配置测试信息：测试者根据具体的测试对象及需求配置测试信息，为测试待测程序做准备，主要包括如下四个子用例：

- **选择测试技术：**测试者根据具体情况选择不同的测试技术。如果待测程序可能具有较多的故障并且故障容易被揭示，简单的随机测试技术是最佳的选择。如果待测程序中的故障难以揭示，适应性蜕变测试技术是最佳的选择。
- **设置测试技术相关的参数：**如果选择适应性蜕变测试技术（以蜕变关系为中心的适应性蜕变测试技术 M-AMT 或以分区为中心的适应性蜕变测试技术 P-AMT），测试者需要设置更新测试剖面的 DRT 和 MDRT 的参数。
- **划分分区：**本文在分区的基础上提出了适应性蜕变测试技术。利用该技术测试待测软件之前，测试者需要对待测软件的输入域划分分区。
- **设置终止条件：**测试系统满足测试者设置的终止条件时停止测试工作。

（2）选择待测程序：测试者选择待测的程序，然后系统根据选择的待测程序自动加载相应的变异体。

- **导入待测程序：**测试者将待测程序导入特定的文件夹后，系统自动加载待测程序。
- **加载变异体：**导入待测程序之后，系统在特点的文件夹下解析、加载待测程序的变异体，并显示在交互界面。

（3）选择待测变异体：系统将待测程序的所有变异体自动地划分为 2 类：传统的变异体和类级别的变异体。

- **选择传统变异体：**在传统变异体界面，测试者可以查看每一个传统变异体的变异体信息并在执行界面选择待测试的传统变异体。
- **选择类级别的变异体：**在类级别的变异体界面，测试者可以查看每一个类级别变异体的变异体信息并在执行界面选择待测试的类级别变异体。

（4）选择待测试的方法：待测程序中的方法通常超过一个，测试者不仅可以选择所有的方法还可以选择特定的方法。

- **选择单个方法：**导入待测程序后，系统解析待测程序并将待测程序所有的方法显示在交互界面。测试者可以选择并测试特定的方法。
- **选择所有方法：**测试者可以选择并测试待测程序所有的方法。

（5）执行测试：执行测试用例（原始测试用例以及衍生测试用例），验证结果是否满足蜕变关系。

- **执行测试用例：**如果测试者选择了随机测试技术，系统随机选择用户输入的测试用例，并根据该测试用例相关的蜕变关系生成相应的衍生测试用例，然后执行测试用例（原始测试用例以及衍生测试用例）。如果测试者选择了适应性蜕变测试技术，系统根据 M-AMT 和 P-AMT 技术选择测试用例与相应的蜕变关系并生成相应的衍生测试用例，最后执行原始测试用例与衍生测试用例。
- **验证测试结果：**验证输出的结果是否违反了蜕变关系。如果测试结果违反了蜕变关系，表明待测软件中存在故障。
- **生成测试报告：**统计并分析测试用例执行通过/失败的信息并生成测试报告。

（6） 查看测试报告：测试者可以查看测试报告，获取全面的测试信息，主要包括：（a）执行的测试用例；（b）输出结果是否违反蜕变关系；（c）杀死的变异体；（d）当前执行的测试用例数目等信息。

4.2 APT2MT 设计与实现

根据上述需求分析，本节详细介绍测试支持工具 APT2MT 的架构、组件的设计与实现。

4.2.1 系统架构

图 4-2 展示了 APT2MT 的系统架构图。选用矩形框表示工具的组件，主要包括：配置测试信息组件、选择待测程序组件、控制测试组件、执行测试组件和验证结果组件，并且每个组件由多个模块构成。所有的组件设计描述如下：

（1） 配置测试信息：负责解析测试用例以及分区相关的文档，设置适应性蜕变测试技术涉及的参数值，设置终止条件。

- **读取文件模块：**主要负责读取并解析特定文件夹下的分区信息，然后将每一个分区的测试用例信息记录在一个列表中并将分区规则显示在交互界面。
- **选择策略模块：**提供 3 种测试技术：（a）随机测试技术（RT）；（b）以蜕变关系为中心的适应性蜕变测试技术（M-AMT）；（c）以分区为中心的适应性蜕变测试技术（P-AMT）。测试者根据实际需求选择一个测试技术。
- **设置终止条件模块：**APT2MT 系统提供两种终止条件：（a）揭示一

个故障；（2）执行一定数目的测试用例。如果选择第二种终止条件，测试者需要设置具体执行测试用例数目的上限。

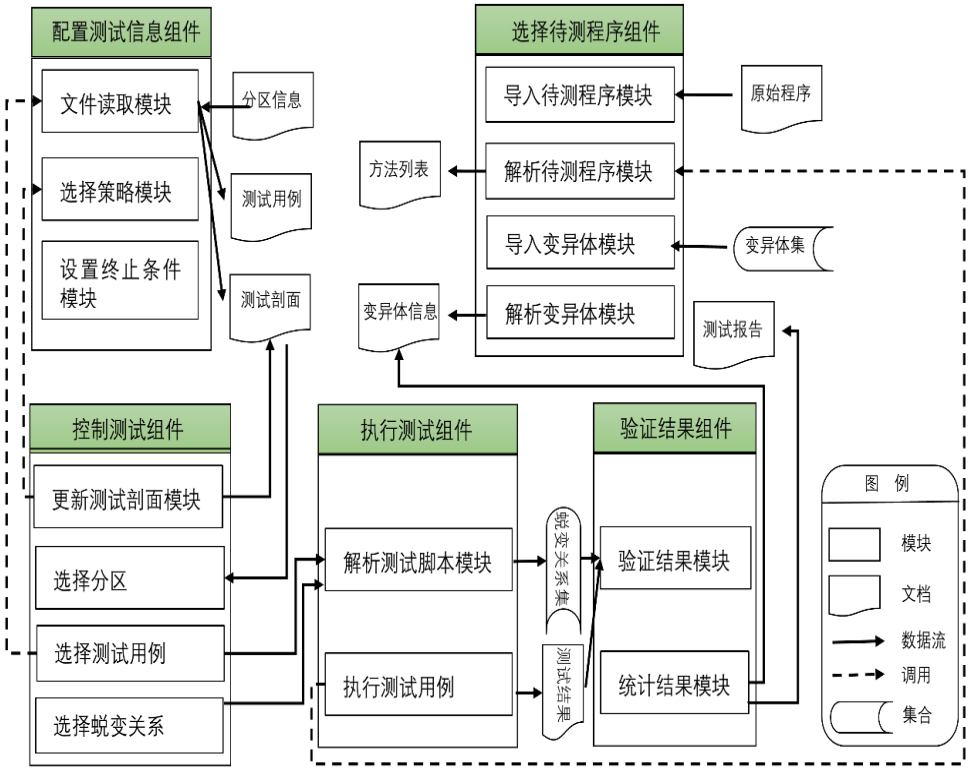


图 4-2 APT2MT 系统架构图

（2） 选择待测程序组件：负责上传、解析待测程序与待测程序的变异体。

- **导入待测程序模块：**测试者将待测程序存入特定的目录下，系统自动识别待测程序，并在交互界面显示待测程序的名字。
- **解析待测程序模块：**系统通过解析待测程序得到待测程序中所有的方法信息，并在交互界面显示所有的方法名字。测试者可以测试一个或者所有的方法。
- **导入变异体模块：**测试者将待测程序的变异体存入特定的目录下，系统自动识别变异体并将变异体划分为传统变异体和类级别的变异体。
- **解析变异体模块：**系统通过解析变异体获取变异体的基本信息，包括：（a）变异体的种类；（b）变异体的数量；（c）每个变异体对应的变异算子；（d）变异体的名称。

（3） 控制测试组件：控制测试的执行过程。

- **更新测试剖面模块：**根据原始测试用例与衍生测试用例的执行结果是

否违反蜕变关系更新测试剖面。如果测试用例的执行结果违反了蜕变关系，原始测试用例与衍生测试用例所在分区的选取概率增大，其它分区的选取概率减小。如果测试用例的执行结果没有违反蜕变关系，原始测试用例与衍生测试用例所在分区的选取概率减小，其它分区的选取概率增大。

- **选择分区模块：**依据上一个模块更新后的测试剖面选择下一个分区。
- **选择测试用例模块：**在上一个模块选取的分区中随机地选择测试用例。
- **选择蜕变关系模块：**根据选取的测试用例以及测试技术选择蜕变关系，生成衍生测试用例。

（4） 执行测试组件：负责执行上一个组件生成的测试用例（原始测试用例与衍生测试用例）。

- **解析测试脚本模块：**原始测试用例、蜕变关系以及衍生测试用例的信息按照固定的格式记录在测试脚本中。系统通过解析测试脚本获取原始测试用例、原始测试用例相关的蜕变关系和原始测试用例以及衍生测试用例所在分区信息。
- **执行测试用例模块：**负责执行测试用例（原始测试用例以及衍生测试用例）。

（5） 验证结果组件：负责验证测试用例的执行结果是否违反了蜕变关系并记录测试信息。

- **验证结果模块：**负责验证测试用例（包括原始测试用例与衍生测试用例）的执行结果是否违反了蜕变关系，然后更新测试剖面模块根据执行结果是否违反蜕变关系更新测试剖面。
- **统计结果模块：**负责收集每一次执行的信息。当系统状态满足终止条件时，该模块处理所有信息并生成测试报告。

4.2.2 APT2MT 实现

介绍适应性蜕变测试支持工具 APT2MT 的实现。

(1) **测试信息获取模块：**主要获取测试者选择的测试技术以及终止条件，并且解析测试帧输入的分区信息，然后将分区以及分区内部测试用例联系起来。

- **Strategy 接口：**本文运用了模板设计模式，设计了 Strategy 接口，其类图如 4-3 所示。所有的测试技术都实现了 Strategy 接口。如果需要添加某种测试技术，只需添加一个 Strategy 接口的实现类，不需要改变

其它模块。

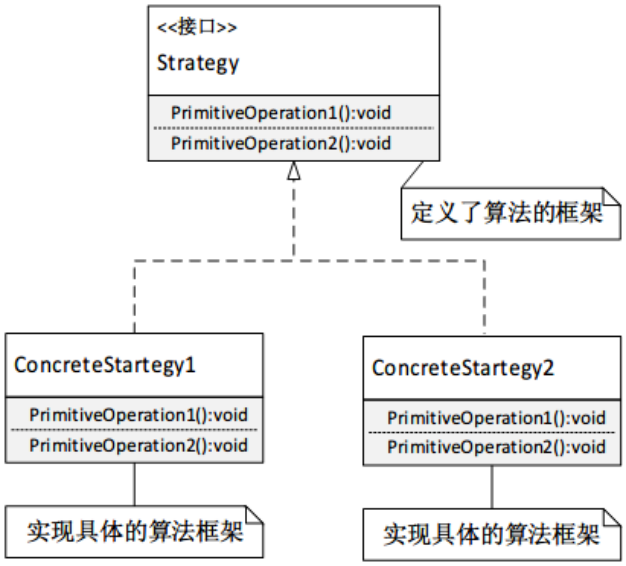


图 4-3 模板模式类图

- **Partition 类：**用于解析测试者在特定文件下存放的分区信息，然后创建一个 Json 格式的数据结构。在该数据结构中，“键”表示分区编号，“值”是一个存放测试用例编号的列表。
- **TestFrameAndMr 类：**用于解析 Junit 测试脚本，然后创建一个 Json 格式的数据结构。在该数据结构中，“键”表示测试用例的编号，“值”是一个存放与该测试用例相关蜕变关系的列表。

(2) **控制测试模块：**依据测试技术的测试原理控制测试过程，包括选择分区、原始测试用例以及蜕变关系。

- **RandomTesting 类：**实现随机测试技术。
- **MAMT 类：**实现了以蜕变关系为中心的适应性蜕变测试技术。
- **PAMT 类：**实现了以分区为中心的适应性蜕变测试技术。

(3) **解析测试脚本模块：**将原始测试用例与蜕变关系建立联系，将分区与蜕变关系建立联系。

- **TestFrameAndMr 类：**本系统利用 Junit 脚本执行测试用例，该脚本按照固定的格式描述了执行的测试用例信息以及相关的蜕变关系。该类用于解析 Junit 测试脚本，然后创建一个 Json 格式的数据结构。在该数据结构中，“键”表示测试用例的编号，“值”是一个存放与该测试用例相关蜕变关系的列表。
- **Configuration 类：**调用 TestFrameAndMr 类和 Partition 类，将分区

与该分区涉及的所有蜕变关系通过 Json 建立联系，“键”表示分区的编号，“值”是分区涉及的所有蜕变关系组成的列表。

(4) **执行测试模块：**执行测试用例，并将结果返回给控制测试模块。

- **TestRun 类：**负责执行测试用例，并且测试的结果信息传递给 Result 类。
- **Result 类：**负责收集测试过程的信息，并且测试结果是否违反蜕变关系的信息传递给测试控制模块。

4.3 系统演示

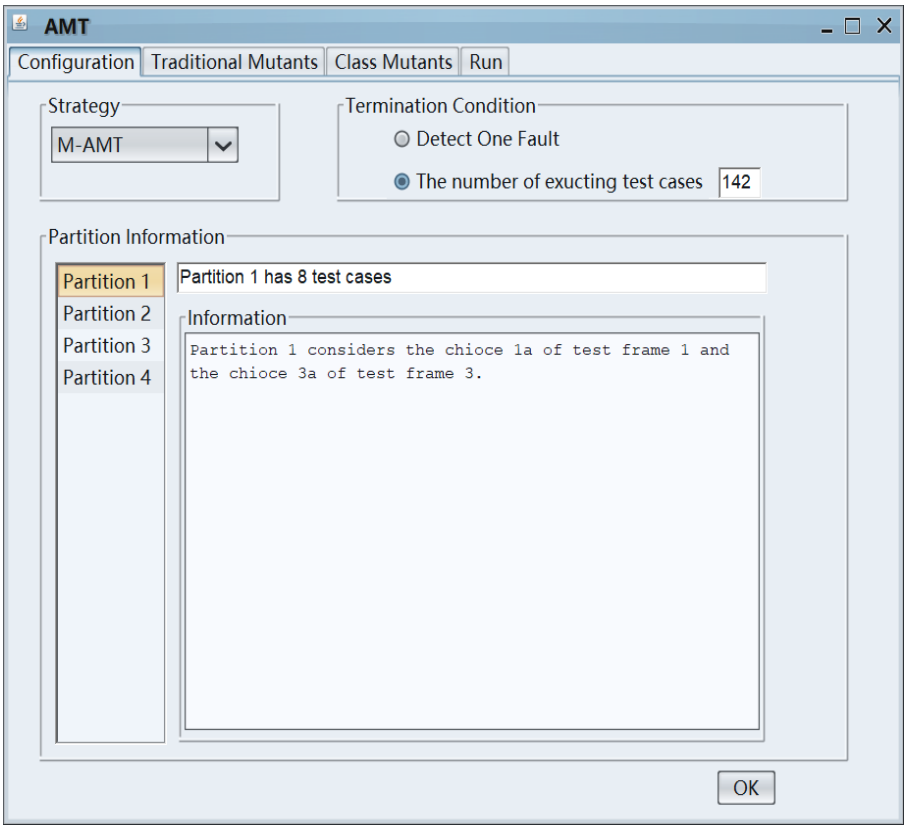


图 4-4 测试信息配置界面

APT2MT 由四个部分组成：测试信息配置界面、测试执行界面、传统变异体界面、类级别的变异体界面。其中，测试信息配置界面提供选择测试技术、设置终止条件和显示分区信息等功能。测试执行界面提供选择待测程序、变异体和待测方法等功能。传统变异体界面显示每一个传统变异算子对应的变异体数目以及每一个变异体具体变异信息。类级别变异体界面显示每一个类级别的变异算子对应的变异体数目以及每一个变异体具体变异信息。

(1) **测试信息配置过程演示：**测试者首先从随机测试（RT）、以蜕变关系

为中心的适应性蜕变测试（M-AMT）和以分区为中心的适应性蜕变测试（P-AMT）三种测试技术中选取一个测试技术，然后选择终止条件。本系统提供两种终止条件：（a）检测一个故障；（b）执行一定数目的测试用例。如果选择第二种终止条件，测试者需要手工输入执行测试用例的数量上限。如果选择 M-AMT 或者 P-AMT，测试者需要将分区以及分区内的测试用例复制到 MuJava[55]的根目录下的“Partitions”文件夹中。测试者点击[OK]按钮之后，系统进入解析分区信息的状态，系统可能保持这种状态一段时间直到完全解析分区信息。然后在[Partition Information]面板的左侧显示分区的数目以及名称。测试者选中某一个分区之后，在[Partition Information]面板的右侧显示选中分区的数目以及分区的规则。在图 4-4 中展示了测试联通计费程序的配置信息，测试者选择了 M-AMT 测试技术，设置了测试用例的执行上限为 100 个。解析分区信息之后，可以看出测试者将软件的输入域划分成 4 个不相交的分区，其中分区 1（Partition 1）有 8 个测试用例，并且该分区考虑了第一个测试帧的 1a 选项和第三个测试帧的 1b 选项。第一个测试帧表示套餐类型，该测试帧有 2 个选项：1a 和 1b。第三个测试帧表示通话时间，包含 2 个选项：3a 和 3b。

(2) **测试执行界面展示：**该界面是该系统的核心，测试者首先选择待测程序，然后选择测试的方法和执行的测试用例，还可以指定测试用例执行的最长时间（单位：秒）。图 4-5 展示了测试联通计费时的界面，测试者指定测试用例最长的执行时间是 3 秒。如果测试用例的执行时间超过 3 秒，则系统认为该测试用例导致程序死锁并认为该测试用例揭示了软件故障。测试者可以选择测试的变异体：传统变异体、类级别的变异体和所有变异体。图中两个明显的列表分别展示了传统变异算子和类级别变异算子的相关信息。文本区域展示了测试结束时，传统变异体和类级别变异体存活和被杀死信息。界面中间的对话框显示当前的测试进度。

4.4 小结

本章详细地讨论了 APT2MT 的设计与实现。该工具支持配置测试信息，设置终止条件，控制测试的过程，执行测试用例并验证测试结果是否违反蜕变关系，有助于提高适应性蜕变测试技术的自动化程度。

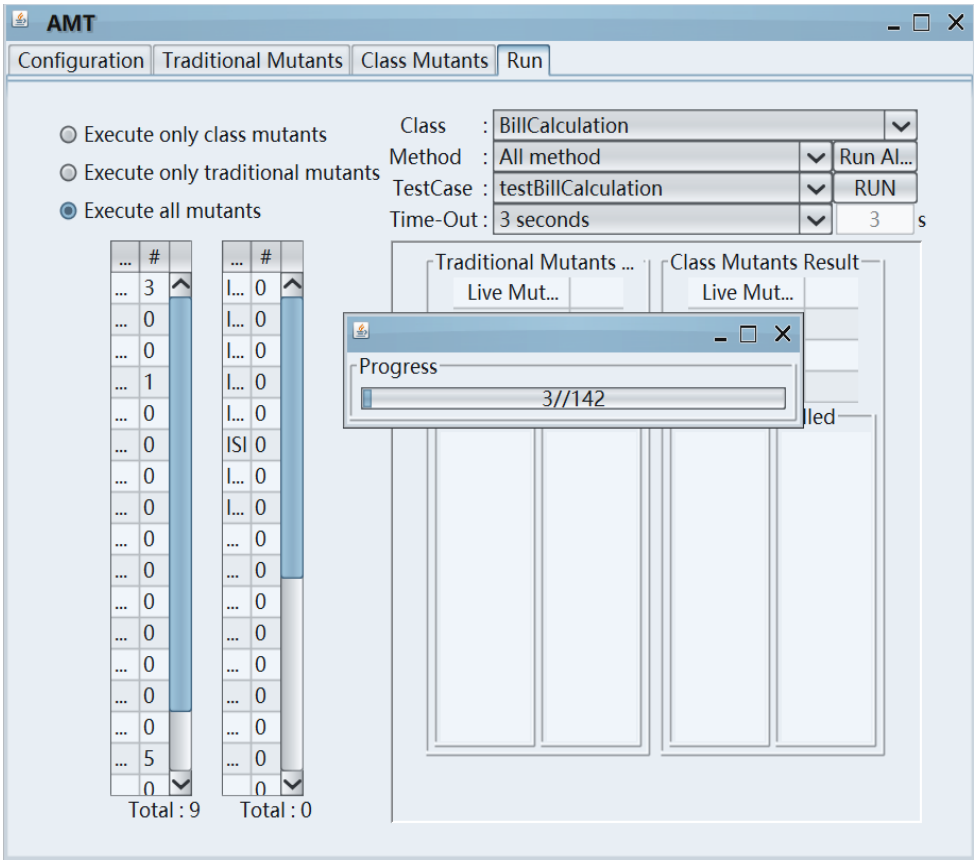


图 4-5 执行测试用例界面

5 经验研究

为了评估提出的适应性蜕变测试的有效性、效率以及支持工具的可行性，本文选择三个实例程序进行实验评估。

5.1 研究问题

围绕以下三个问题进行试验评估：

(1) **以蜕变关系为中心的适应性蜕变测试技术（M-AMT）的故障检测效率：**评估 M-AMT 在三个实例程序上的故障检测率。

(2) **以分区为中心的适应性蜕变测试技术（P-AMT）的故障检测效率：**评估 P-AMT 在三个实例程序上的故障检测率。

(3) **M-AMT 和 P-AMT 选择测试用例的性能：**评估 M-AMT 和 P-AMT 在三个实例程序上选择测试用例的时间开销。

5.2 实验对象

本文采用联通计费服务（PBS）、航空行李托运服务（ABBS）和费用补偿服务（EXP）实例程序作为研究对象。

表 5-1 PBS 服务输入规格描述

输入参数	参数类型
套餐类型（PlanType）	String
月租（PlanFee）	Int
通话时间（TalkTime）	Int
使用的流量（Flow）	Int
可视电话时长（CallViewTime）	Int

5.2.1 联通计费服务（PBS）

联通计费服务是一个 Web 服务程序，根据联通 3G 网络的两种基本套餐进行手机业务计费。该服务根据选取的套餐类型、月租费用、通话时间以及使用的流量计算客户需要交纳的话费金额（单位：CNY）。PBS 系统的输入规则如表 5-1 所示。考虑到国内基本不使用可视通话业务，没有将可视电话费用添加到总费用中。PBS 系统的输出参数为 Bill，类型是 Double，计算公式如下：

$$Bill = (TalkTime - TalkTimeBench) \times TalkTimePer + (Flow - FlowBench) \times FlowPer + PlanFee \quad (14)$$

其中，TalkTime 参数是客户当月的通话时间，TalkTimePer 参数是每分钟的通话费用，FlowPer 参数是 1M 流量的费用，TalkTimeBench 参数和 FlowBench 参数是免费的通话时间和流量，PlanFee 参数是客户每月应缴纳的月租费用。表 5-2 展示了参数的详细信息。

表 5-2 PBS 系统计费标准

PlanType	PlanFee（CNY）	TalkTimeBench（min）	FlowBench（M）	TalkTimePer（CNY）	FlowPer（CNY）
A	46	50	150	0.25	0.3
	96	96	240	0.15	0.3
	286	286	900	0.15	0.3
	886	3000	3000	0.15	0.3
B	46	120	40	0.25	0.3
	96	450	80	0.15	0.3
	126	680	100	0.15	0.3
	186	1180	150	0.15	0.3

5.2.2 航空行李托运服务（ABBS）

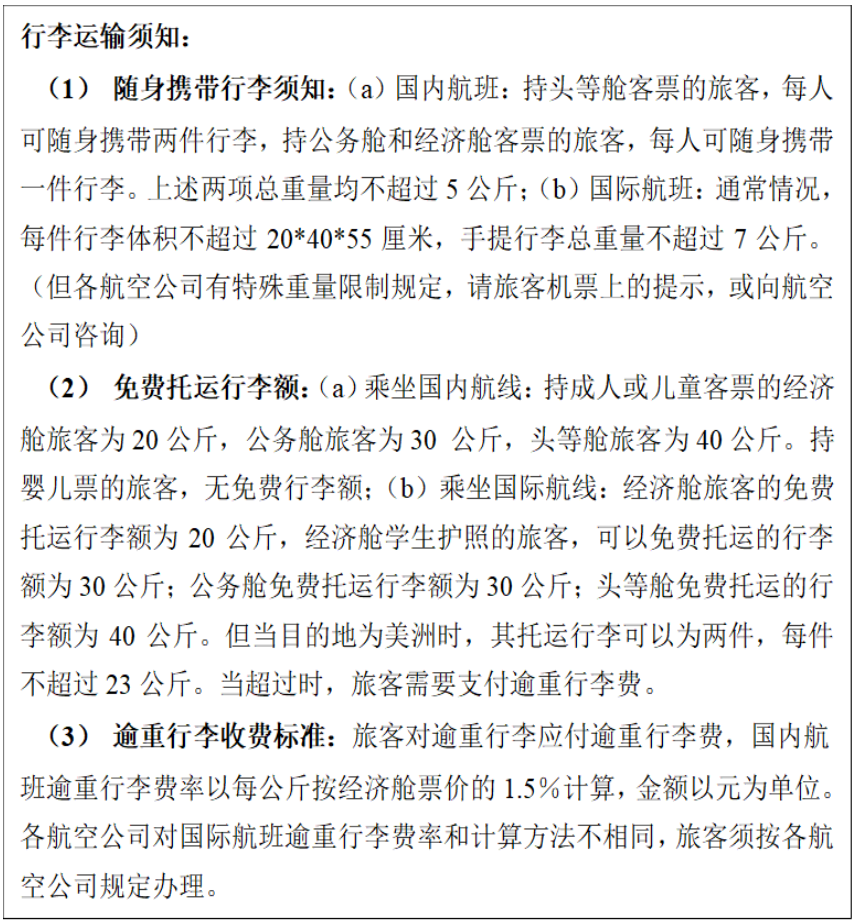
航空行李托运服务（ABBS）的程序实现参照了北京首都国际机场关于行李托运的相关规定，详细的信息如图 5-1 所示。该服务实现了查询可随身携带的行李重量（Hand carry）、查询可免费托运的行李重量（Checked-in for free）以及查询托运费用的功能。本文只对查询托运费用的功能进行经验研究。该功能的输入规则如表 5-3。

表 5-3 ABBS 服务输入规格描述

输入参数	参数类型	输入限制
座舱等级（AirClass）	Int	{0, 1, 2, 3}
航班类型（Region）	Int	{0, 1}
是否是学生（IsStudent）	Boolean	{true, false}
行李重量（Luggage）	Double	> 0
托运费（Airfare）	Double	> 0

AirClass 参数表示座舱等级，可以被赋值 0、1、2、3，分别代表头等舱、商务舱、经济舱和婴儿票。Region 参数表示航班类型，可以被赋值 0、1，分别表示国内航班和国际航班。IsStudent 参数表示乘客是否为学生，“true”表示乘客是学生，“false”表示乘客不是学生。Luggage 表示乘客携带的行李重

量。Airfare 参数表示乘客需支付的托运费。



行李运输须知：

（1） 随身携带行李须知：（a）国内航班：持头等舱客票的旅客，每人可随身携带两件行李，持公务舱和经济舱客票的旅客，每人可随身携带一件行李。上述两项总重量均不超过 5 公斤；（b）国际航班：通常情况，每件行李体积不超过 20*40*55 厘米，手提行李总重量不超过 7 公斤。（但各航空公司有特殊重量限制规定，请旅客机票上的提示，或向航空公司咨询）

（2） 免费托运行李额：（a）乘坐国内航线：持成人或儿童客票的经济舱旅客为 20 公斤，公务舱旅客为 30 公斤，头等舱旅客为 40 公斤。持婴儿票的旅客，无免费行李额；（b）乘坐国际航线：经济舱旅客的免费托运行李额为 20 公斤，经济舱学生护照的旅客，可以免费托运的行李额为 30 公斤；公务舱免费托运行李额为 30 公斤；头等舱免费托运的行李额为 40 公斤。但当目的地为美洲时，其托运行李可以为两件，每件不超过 23 公斤。当超过时，旅客需要支付逾重行李费。

（3） 逾重行李收费标准：旅客对逾重行李应付逾重行李费，国内航班逾重行李费率以每公斤按经济舱票价的 1.5%计算，金额以元为单位。各航空公司对国际航班逾重行李费率和计算方法不相同，旅客须按各航空公司规定办理。

图 5-1 北京首都国际机场关于行李托运的相关规定

计算托运费用的方法输出的参数是 Luggagefee，计算的公式如下：

$$Luggagefee = (Luggage - Benchluggage) \times Airfare \times 1.5\% \quad (15)$$

其中，Luggagefee 表示乘客必须支付的费用。Benchluggage 表示免费托运的行李重量，系统根据乘客的航班类型（国内航班或者国际航班）、座舱等级和身份（是否为学生）选择恰当的 Benchluggage 值，表 5-4 展示了不同情况下 Benchluggage 的取值。如果行李的重量小于免费托运的重量，乘客不需要支付托运费用。当乘客选择了国际航班并且为学生时，Benchluggage 的值为 30kg。

5.2.3 费用补偿系统（EXP）

费用补偿系统协助公司销售总监处理以下业务：（a）收取高级销售经理和销售经理使用公司车辆并超出规定英里数应向公司缴纳的费用；（b）处理高级销售经理、销售经理和销售主管的机票、酒店住宿、饮食以及电话费用

的补偿请求。公司有如下规定：（a）只有高级销售经理和销售经理可以使用公司的车辆；（b）当销售额达到$100,000 时，高级销售经理和销售经理可以报销其它种类的费用。该系统输入规格的详细说明如表 5-5。

表 5-4 Benchluggage 取值说明

座舱（AirClass）	免费托运的重量（Benchluggage）
0	40kg
1	30kg
2	20kg
3	0kg

表 5-5 EXP 系统输入规格描述

输入参数	类型
工号（Staff ID）	String
姓名（Staff name）	String
职称（Staff level）	Int
实际英里数（Actual mileage for the month）	Double
销售额（Monthly sales amount）	Double
机票费用（Airfar）	Double
其它费用（Other expenses）	Double

表 5-5 中，Staff ID 表示员工的工号，Staff name 表示员工的姓名，Staff level 表示员工的职称（高级销售经理（senior sales manager）、销售经理（sales manager）和销售主管（sales supervisor）），Actual mileage for the month 表示使用公司的车辆行驶的英里数，Monthly sales amount 表示实际的月销售额，Airfar 表示机票费用，Other expenses 表示月销售额达到$100,000 时，可以额外报销的金额。

EXP 系统输出某一个员工当月的补偿费用信息，包括：工号、姓名、职称以及补偿金额。该系统输出的详细说明如表 5-6。

表 5-6 中，Staff ID、Staff name 和 Staff level 参数表示的含义与表 5-5 相同。Total reimbursement amount 表示某个员工当月的补偿金。

EXP 系统根据员工的职称选择不同的方式计算每月的补偿金，详细算法如表 5-7。当月销售额到达表 5-8 所示的条件时，员工可以享受飞机票补贴服务。不同职称的员工计算偿还公司额外英里费用的方式不同，如表 5-9。

表 5-6 EXP 系统输出规格描述

输出参数	类型
工号（Staff ID）	String
姓名（Staff name）	String
职称（Staff level）	Int
补偿金（Total reimbursement amount）	Double

表 5-7 EXP 系统补偿费计算方式

职称	计算方式
高级销售经理和销售经理	补偿费=机票费用+其它-额外公里费用
销售主管	补偿费=机票费用+其它

表 5-8 补贴飞机票的条件

职称	条件
高级销售经理	不补偿
销售经理	月销售额>=$50,000
销售主管	月销售额>=$80,000

表 5-9 额外英里费计算方式

职称	限额（x）	实际英里数（y）	计算公式	补偿条件
高级销售经理	4000	>0	$5\times(y-x)$	$y>x$
销售经理	3000	>0	$8\times(y-x)$	$y>x$

5.3 实验设置

5.3.1 待评估的测试技术

本文提出的 M-AMT 和 P-AMT 作为研究的对象。为了突出蜕变测试与适应性分区测试集成的有效性，本文评估与比较 M-AMT、P-AMT 和传统蜕变测试的故障检测的有效性与效率。

5.3.2 度量标准

评估某个测试技术有效性的常见度量指标包括：（a）P-measure（在当前测试用例集中至少检测一个故障的概率）、E-measure（当前测试用例集期望检测的故障数目）、F-measure（检测一个故障需要的测试用例数目）、T-measure（检测所有的故障需要的测试用例数目）。传统蜕变测相关研究中，通常采用

F-measure 评估蜕变测试技术的有效性，为了便于评估与比较本文将 F-measure 作为评估所提测试技术有效性的指标之一。

另外，测试资源是有限的，测试者不可能执行测试用例集中所有的测试用例。本文控制不同测试技术执行相同数目的测试用例测试同一个对象，比较它们揭示故障的数目。明显地，揭示的故障数目越多，测试技术揭示故障的能力就越强。本文提出一个新的度量指标 N-measure，表示测试技术执行一定数目测试用例揭示故障的数目。本文计划执行一半的测试用例，然后比较测试技术揭示故障的数目，实验过程中发现，ABBS 和 EXP 系统执行一半的测试用例时，三种测试技术均“杀死”了所有变异体。为了显示测试技术执行相同数目的测试用例揭示故障数目的差异，N-measure 指标设置如下：（a）PBS 系统执行测试用例集中一半的测试用例（70 个）；（b）ABBS 系统执行测试用例集中 1/4 的测试用例（200 个）；（c）EXP 系统执行测试用例集中 1/5 的测试用例（200 个）。

对于研究问题（3），采用 Time-measure 表示选择测试用例的时间。

此外，为了保证评估结果的可靠性，我们重复实验 20 次，统计不同度量指标（F-measure、N-measure 和 Time-measure）的平均值。

5.3.3 蜕变关系识别

本文利用 METRIC 技术识别三个待测程序的蜕变关系，识别待测程序蜕变关系的过程描述如下：

{1a,2a,3a,4a}	{1a,2a,3a,4b}	{1a,2a,3b,4a}	{1a,2a,3b,4b}
{1a,2b,3a,4a}	{1a,2b,3a,4b}	{1a,2b,3b,4a}	{1a,2b,3b,4b}
{1a,2e,3a,4a}	{1a,2e,3a,4b}	{1a,2e,3b,4a}	{1a,2e,3b,4b}
{1a,2f,3a,4a}	{1a,2f,3a,4b}	{1a,2f,3b,4a}	{1a,2f,3b,4b}
{1b,2a,3a,4a}	{1b,2a,3a,4b}	{1b,2a,3b,4a}	{1b,2a,3b,4b}
{1b,2b,3a,4a}	{1b,2b,3a,4b}	{1b,2b,3b,4a}	{1b,2b,3b,4b}
{1b,2c,3a,4a}	{1b,2c,3a,4b}	{1b,2c,3b,4a}	{1b,2c,3b,4b}
{1b,2d,3a,4a}	{1b,2d,3a,4b}	{1b,2d,3b,4a}	{1b,2d,3b,4b}

图 5-2 PBS 系统所有完整测试帧

(1) **识别 PBS 系统的蜕变关系：**根据 4.2.1 章节展示的 PBS 系统输入规格说明，测试者首先识别影响软件行为的输入参数或者环境条件。系统的输入参数或者环境条件被定义成范畴（Category），每一个范畴可以划分若干不相交的子集，这些子集称为选项（Choice）。PBS 系统的范畴与选项信息展示在表 5-10 中。为了方便描述，每一个范畴的选项用 NX 的形式表示，其中 N 为

阿拉伯数字，X 为英文字母，例如套餐范畴的两个选项：A 套餐和 B 套餐分别表示为 1a 和 1b。该系统划分了 4 个范畴，每个范畴对应的选项个数依次是 2，6，2，2。然而，完整测试帧的个数并不是 48（$2 \times 6 \times 2 \times 2$），原因是其中一些选项组合是无效的。在表 5-2 中，套餐 A 没有月租为 126 元的业务，因此测试帧$\{1a, 2c, 3a, 4a\}$是无效的选项组合。图 5-2 展示了 PBS 系统所有的完整测试帧。本文利用 METRIC 技术成对地比较完整测试帧，识别了 142 个蜕变关系。表 5-11 展示了部分蜕变关系，其中第二列“→”的左边表示第一个测试帧，右边表示第二个测试帧。$bill_1$表示第一个测试帧对应的输出，$bill_2$表示第二个测试帧对应的输出。

表 5-10 PBS 系统范畴和选项信息

范畴（Category）	选项（Choice）
套餐（PlanType）	A 套餐（1a）
	B 套餐（1b）
月租（PlanFee）	46CNY（2a）
	96CNY（2b）
	126CNY（2c）
	186CNY（2d）
	286CNY（2e）
	886CNY（2f）
通话时间（TalkTime）	talkTime < talkTimeBench（3a）
	talkTime >= talkTimeBench（3b）
流量（Flow）	flow < flowBench（4a）
	flow >= flowBench（4b）

表 5-11 PBS 系统部分蜕变关系

序号	输入之间的关系（r）	输出之间的关系（rf）
1	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2a, 3a, 4b\}$	$bill_1 > bill_2$
2	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2a, 3b, 4a\}$	$bill_1 > bill_2$
3	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2a, 3b, 4b\}$	$bill_1 > bill_2$
4	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2b, 3a, 4a\}$	$bill_1 > bill_2$
5	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2b, 3a, 4b\}$	$bill_1 > bill_2$
6	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2b, 3b, 4b\}$	$bill_1 > bill_2$
7	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2b, 3b, 4b\}$	$bill_1 > bill_2$
8	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2e, 3a, 4a\}$	$bill_1 > bill_2$
9	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2e, 3a, 4b\}$	$bill_1 > bill_2$
10	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2e, 3b, 4a\}$	$bill_1 > bill_2$
11	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2e, 3b, 4b\}$	$bill_1 > bill_2$
12	$\{1a, 2a, 3a, 4a\} \rightarrow \{1a, 2a, 3a, 4b\}$	$bill_1 > bill_2$

{1a, 2a, 3a, 4a, 5a}	{1a, 2a, 3a, 4a, 5b}	{1a, 2a, 3a, 4b, 5a}	{1a, 2a, 3a, 4b, 5b}
{1a, 2b, 3a, 4a, 5a}	{1a, 2b, 3a, 4a, 5b}	{1a, 2b, 3a, 4b, 5a}	{1a, 2b, 3a, 4b, 5b}
{1b, 2a, 3a, 4a, 5a}	{1b, 2a, 3a, 4a, 5b}	{1b, 2a, 3a, 4b, 5a}	{1b, 2a, 3a, 4b, 5b}
{1b, 2b, 3a, 4a, 5a}	{1b, 2b, 3a, 4a, 5b}	{1b, 2b, 3a, 4b, 5a}	{1b, 2b, 3a, 4b, 5b}
{1c, 2a, 3a, 4a, 5a}	{1c, 2a, 3a, 4a, 5b}	{1c, 2a, 3a, 4b, 5a}	{1c, 2a, 3a, 4b, 5b}
{1c, 2b, 3a, 4a, 5a}	{1c, 2b, 3a, 4a, 5b}	{1c, 2b, 3a, 4b, 5a}	{1c, 2b, 3a, 4b, 5b}
{1c, 2a, 3a, 4a, 5a}	{1c, 2a, 3b, 4a, 5b}	{1c, 2a, 3b, 4b, 5a}	{1c, 2a, 3b, 4b, 5b}
{1c, 2b, 3b, 4a, 5a}	{1c, 2b, 3b, 4a, 5b}	{1c, 2b, 3b, 4b, 5a}	{1c, 2b, 3b, 4b, 5b}
{1d, 2a, 3a, 4a, 5a}	{1d, 2a, 3a, 4a, 5b}	{1d, 2a, 3a, 4b, 5a}	{1d, 2a, 3a, 4b, 5b}
{1d, 2b, 3a, 4a, 5a}	{1d, 2b, 3a, 4a, 5b}	{1d, 2b, 3a, 4b, 5a}	{1d, 2b, 3a, 4b, 5b}

图 5-3 ABBS 系统所有完整测试帧

表 5-12 ABBS 系统范畴和选项信息

范畴（Category）	选项（Choice）
座舱等级（AirClass）	头等舱（1a）
	商务舱（1b）
	经济舱（1c）
	婴儿票（1d）
航班类型（Region）	国内航班（2a）
	国际航班（2b）
乘客身份（IsStudent）	不是学生（3a）
	是学生（3b）
行李重量（Luggage）	luggage < luggageBench（4a）
	luggage >= luggageBench（4b）
托运费（Airfare）	airfare = 0（5a）
	airfare > 0（5b）

(2) **识别 ABBS 系统的蜕变关系：**根据 4.2.2 章节展示的 ABBS 系统输入规格说明，测试者首先识别影响软件行为的输入参数或者环境条件。系统的输入参数或者环境条件被定义成范畴（Category），每一个范畴可以划分若干不相交的子集，这些子集称为选项（Choice）。ABBS 系统的范畴与选项信息展示在表 5-12 中。为了方便描述，每一个范畴的选项用 NX 的形式表示，其中 N 为阿拉伯数字，X 为英文字母，例如航班类型范畴的两个选项：国内航班和国际航班分别表示为 2a 和 2b。该系统划分了 5 个范畴，每个范畴对应的选项个数依次是 4，2，2，2，2。然而，完整测试帧的个数并不是 64（$4 \times 2 \times 2 \times 2 \times 2$），原因是其中一些选项组合是无效的。ABBS 系统的完整测试帧信息如图 5-3。本文利用 METRIC 技术成对地比较完整测试帧，识别了 735 个蜕变关系。表 5-13 展示了部分蜕变关系，其中第二列“→”的左边表示第一个测试帧，右边表示第二个测试帧。$luggagefee_1$表示第一个测试帧对应的

输出，$luggagefee_2$表示第二个测试帧对应的输出。

{1a, 2a, 3a, 4a, 5a}	{1a, 2a, 3a, 4a, 5b}	{1a, 2a, 3a, 4b, 5a}	{1a, 2a, 3a, 4b, 5b}
{1a, 2b, 3a, 4a, 5a}	{1a, 2b, 3a, 4a, 5b}	{1a, 2b, 3a, 4b, 5a}	{1a, 2b, 3a, 4b, 5b}
{1b, 2a, 3a, 4a, 5a}	{1b, 2a, 3a, 4a, 5b}	{1b, 2a, 3a, 4b, 5a}	{1b, 2a, 3a, 4b, 5b}
{1b, 2b, 3a, 4a, 5a}	{1b, 2b, 3a, 4a, 5b}	{1b, 2b, 3a, 4b, 5a}	{1b, 2b, 3a, 4b, 5b}
{1c, 2a, 3a, 4a, 5a}	{1c, 2a, 3a, 4a, 5b}	{1c, 2a, 3a, 4b, 5a}	{1c, 2a, 3a, 4b, 5b}
{1c, 2b, 3a, 4a, 5a}	{1c, 2b, 3a, 4a, 5b}	{1c, 2b, 3a, 4b, 5a}	{1c, 2b, 3a, 4b, 5b}
{1c, 2a, 3a, 4a, 5a}	{1c, 2a, 3b, 4a, 5b}	{1c, 2a, 3b, 4b, 5a}	{1c, 2a, 3b, 4b, 5b}
{1c, 2b, 3b, 4a, 5a}	{1c, 2b, 3b, 4a, 5b}	{1c, 2b, 3b, 4b, 5a}	{1c, 2b, 3b, 4b, 5b}
{1d, 2a, 3a, 4a, 5a}	{1d, 2a, 3a, 4a, 5b}	{1d, 2a, 3a, 4b, 5a}	{1d, 2a, 3a, 4b, 5b}
{1d, 2b, 3a, 4a, 5a}	{1d, 2b, 3a, 4a, 5b}	{1d, 2b, 3a, 4b, 5a}	{1d, 2b, 3a, 4b, 5b}

图 5-4 EXP 系统所有完整测试帧

(3) **识别 EXP 系统的蜕变关系：**根据 4.2.3 章节展示的 EXP 系统输入规格说明，测试者首先识别影响软件行为的输入参数或者环境条件。系统的输入参数或者环境条件被定义成范畴（Category），每一个范畴可以划分若干不相交的子集，这些子集称为选项（Choice）。EXP 系统的范畴与选项信息展示在表 5-14 中。为了方便描述，每一个范畴的选项用 NX 的形式表示，其中 N 为阿拉伯数字，X 为英文字母，例如航班类型范畴的两个选项：国内航班和国际航班分别表示为 2a 和 2b。该系统划分了 5 个范畴，每个范畴对应的选项个数依次是 3，3，4，2，2。然而，完整测试帧的个数并不是 144（$3 \times 3 \times 4 \times 2 \times 2$），原因是其中一些选项组合是无效的。EXP 系统的完整测试帧信息如图 5-4。本文利用 METRIC 技术成对地比较完整测试帧，识别了 1130 个蜕变关系。表 5-15 展示了部分蜕变关系，其中第二列“→”的左边表示第一个测试帧，右边表示第二个测试帧。$amount_1$表示第一个测试帧对应的输出，$amount_2$表示第二个测试帧对应的输出。

表 5-13 ABBS 系统部分蜕变关系

序号	输入之间的关系（r）	输出之间的关系（rf）
1	{1a, 2a, 3a, 4a, 5a} → {1a, 2a, 3a, 4a, 5b}	$luggagefee_1 = luggagefee_2 = 0$
2	{1a, 2a, 3a, 4a, 5a} → {1a, 2a, 3a, 4b, 5a}	$luggagefee_1 = luggagefee_2 = 0$
3	{1a, 2a, 3a, 4a, 5a} → {1a, 2b, 3a, 4a, 5a}	$luggagefee_1 = luggagefee_2 = 0$
4	{1a, 2a, 3a, 4a, 5a} → {1b, 2a, 3a, 4a, 5a}	$luggagefee_1 = luggagefee_2 = 0$
5	{1a, 2a, 3a, 4a, 5a} → {1c, 2a, 3a4a, 5a}	$luggagefee_1 = luggagefee_2 = 0$
6	{1a, 2a, 3a, 4a, 5a} → {1d, 2a, 3a, 4a, 5a}	$luggagefee_1 = luggagefee_2 = 0$
7	{1a, 2a, 3a, 4a, 5b} → {1a, 2a, 3a, 4b, 5b}	$luggagefee_1 < luggagefee_2$
8	{1a, 2a, 3a, 4a, 5b} → {1a, 2b, 3a, 4a, 5b}	$luggagefee_1 = luggagefee_2 = 0$
9	{1a, 2a, 3a, 4a, 5b} → {1b, 2a, 3a, 4a, 5b}	$luggagefee_1 = luggagefee_2 = 0$

表 5-13 ABBS 系统部分蜕变关系（续）

序号	输入之间的关系（r）	输出之间的关系（rf）
10	{1a, 2a, 3a, 4a, 5b} → {1c, 2a, 3a, 4a, 5b}	$luggagefee_1 = luggagefee_2 = 0$
11	{1a, 2a, 3a, 4a, 5b} → {1d, 2a, 3a, 4a, 5b}	$luggagefee_1 = luggagefee_2 = 0$
12	{1a, 2a, 3a, 4b, 5a} → {1d, 2a, 3a, 4b, 5b}	$luggagefee_1 < luggagefee_2$

表 5-14 EXP 系统范畴与选项信息

范畴（Category）	选项（Choice）
职称（Staff Level）	高级销售经理（1a）
	销售经理（1b）
	销售主管（1c）
实际英里数（Mileage）	$0 \leq$ Mileage ≤ 3000（2a）
	$3000 <$ Mileage ≤ 4000（2b）
	Mileage > 3000（2c）
月销售额（Amounts）	$0 \leq$ Amounts $< 50{,}000$（3a）
	$50{,}000 \leq$ Amounts $< 80{,}000$（3b）
	$80{,}000 \leq$ Amounts $< 100{,}000$（3c）
	$0 \leq$ Amounts $\geq 100{,}000$（3d）
飞机票（Airfare）	Airfare $= 0$（4a）
	Airfare > 0（4b）
其它费用（Expense）	Expense $= 0$（5a）
	Expense > 0（5b）

表 5-15 EXP 系统部分蜕变关系

序号	输入之间的关系（r）	输出之间的关系（rf）
1	{1a, 2a, 3a, 4a} → {1a, 2a, 3a, 4b}	$amount_1 < amount_2$
2	{1a, 2a, 3a, 4a} → {1a, 2a, 3b, 4a}	$amount_1 = amount_2 = 0$
3	{1a, 2a, 3a, 4a} → {1a, 2a, 3c, 4a}	$amount_1 = amount_2 = 0$
4	{1a, 2a, 3a, 4a} → {1a, 2b, 3a, 4a}	$amount_1 = amount_2 = 0$
5	{1a, 2a, 3a, 4a} → {1a, 2c, 3a, 4a}	$amount_1 > amount_2$
6	{1a, 2a, 3a, 4a} → {1a, 2a, 3a, 4a}	$amount_1 = amount_2 = 0$
7	{1a, 2a, 3b, 4a} → {1a, 2a, 3b, 4b}	$amount_1 < amount_2$
8	{1a, 2a, 3b, 4a} → {1a, 2a, 3c, 4a}	$amount_1 = amount_2 = 0$
9	{1a, 2a, 3b, 4a} → {1a, 2b, 3b, 4a}	$amount_1 = amount_2 = 0$
10	{1a, 2a, 3b, 4a} → {1a, 2c, 3b, 4a}	$amount_1 > amount_2$
11	{1a, 2a, 3b, 4a} → {1b, 2a, 3b, 4a}	$amount_1 = amount_2 = 0$
12	{1a, 2a, 3c, 4a} → {1a, 2a, 3c, 4b}	$amount_1 < amount_2$

{1a, 2a, 3a, 4a}	{1a, 2a, 3a, 4b}	{1a, 2a, 3b, 4a}	{1a, 2a, 3b, 4b}
{1a, 2a, 3c, 4a}	{1a, 2a, 3c, 4b}	{1a, 2a, 3d, 4a, 5a}	{1a, 2a, 3d, 4a, 5b}
{1a, 2a, 3d, 4b, 5a}	{1a, 2a, 3d, 4b, 5b}	{1a, 2b, 3a, 4a}	{1a, 2b, 3a, 4b}
{1a, 2b, 3b, 4a}	{1a, 2b, 3b, 4b}	{1a, 2b, 3c, 4a}	{1a, 2b, 3c, 4b}
{1a, 2b, 3d, 4a, 5a}	{1a, 2b, 3d, 4a, 5b}	{1a, 2b, 3d, 4b, 5a}	{1a, 2b, 3d, 4b, 5b}
{1a, 2c, 3a, 4a}	{1a, 2c, 3a, 4b}	{1a, 2c, 3b, 4a}	{1a, 2c, 3b, 4b}
{1a, 2c, 3c, 4a}	{1a, 2c, 3c, 4b}	{1a, 2c, 3d, 4a, 5a}	{1a, 2c, 3d, 4a, 5b}
{1a, 2c, 3d, 4b, 5a}	{1a, 2c, 3d, 4b, 5b}	{1b, 2a, 3a, 4a}	{1b, 2a, 3a, 4b}
{1b, 2a, 3b, 4a}	{1b, 2a, 3b, 4b}	{1a, 2a, 3c, 4a}	{1b, 2a, 3c, 4b}
{1b, 2a, 3d, 4a, 5a}	{1b, 2a, 3d, 4a, 5b}	{1a, 2a, 3d, 4b, 5a}	{1b, 2a, 3b, 4b, 5b}
{1b, 2b, 3a, 4a}	{1b, 2b, 3a, 4b}	{1a, 2b, 3b, 4a}	{1b, 2b, 3b, 4b}
{1b, 2b, 3c, 4a}	{1b, 2b, 3c, 4b}	{1a, 2b, 3d, 4a, 5a}	{1b, 2b, 3d, 4a, 5b}
{1b, 2b, 3d, 4b, 5a}	{1b, 2b, 3d, 4b, 5b}	{1a, 2c, 3a, 4a}	{1b, 2c, 3a, 4b}
{1b, 2c, 3b, 4a}	{1b, 2c, 3b, 4b}	{1a, 2c, 3c, 4a}	{1b, 2c, 3c, 4b}
{1b, 2c, 3d, 4a, 5a}	{1b, 2c, 3d, 4a, 5b}	{1a, 2c, 3d, 4b, 5a}	{1b, 2c, 3d, 4b, 5b}
{1c, 3a, 4a}	{1c, 3a, 4b}	{1c, 3b, 4a}	{1c, 3b, 4b}
{1c, 3c, 4a}	{1c, 3c, 4b}	{1c, 3d, 4a, 5a}	{1c, 3d, 4a, 5b}
{1c, 3d, 4b, 5a}	{1c, 3d, 4b}		

图 5-4 EXP 系统所有完整测试帧

5.3.4 测试剖面设置

测试者根据具体的测试对象设置初始测试剖面。均等的概率分布作为初始测试剖面是一个保守、可行的设置初始剖面的方案。另一方面，测试者也可以根据以往的测试经验，增大某些分区的选取概率并且减小其它分区的选取概率。本文将均等的概率分布作为所有实验对象的初始测试剖面。

5.3.5 分区设置

测试帧是一类测试用例的抽象并且这些测试用例具有相似的行为，即测试帧具有同构的性质。测试帧的同构性是指如果该测试帧实例化的一个测试用例能够揭示软件的故障，那么该测试帧实例化的其它测试用例也能揭示软件中故障；反之，如果一个测试用例不能揭示软件中的故障，那么其它测试用例也不能揭示软件中的故障。因此本文针对一个测试帧实例化一个测试用例。

事实上，每个完整测试帧可以作为一个分区，但是这种细粒度的分区方式导致每一个分区只有一个测试用例。在这种情况下，本文提出的测试技术不能很好地测试待测程序。

本文采用一种粗粒度的分区方式：只考虑两个范畴，并有效地组合范畴

中的选项，每一种选项组合作为一个分区。例如，本文考虑 PBS 系统中的套餐和通话时间这两个范畴，根据表 5-10 可以观察到两个范畴的选项分别为$\{1a, 1b\}$和$\{3a, 3b\}$，因此可以将待测软件的输入域划分为 4（2×2）分区。表 5-19 展示了三个实验对象的分区信息。

表 5-16 实验对象的分区信息

实验对象	范畴	分区数目
PBS	套餐（PlanType）和通话时间（TalkTime）	4
ABBS	座舱等级（AirClass）和航班类型（Region）	8
EXP	职称（StaffLevel）和月销售额（Amounts）	12

5.3.6 参数设置

本文涉及的自变量中，需要在测试任务开始前设置 M-AMT 以及 P-AMT 中的 DRT 以及 MDRT 的参数。本文参考文献[46-48]，设置 DRT 以及 MDRT 的参数值为 0.05。

5.3.7 实验环境

本文使用 Java 语言编写测试脚本，运行在虚拟机中的 Ubuntu16.04 64 位操作系统上。该系统有 2 个 CPU 和 2GB 内存。

5.3.8 变异体

本文利用变异分析工具 Mujava[55]分别为 PBS 程序、ABBS 程序和 EXP 系统生成了 210，187，180 个变异体。首先识别了每一个实验对象的等价变异体，然后用随机测试技术检测每一个变异体被杀死需要的测试用例数目，挑选至少需要 20 个测试用例才能杀死的变异体。三个实验对象的变异体信息如表 5-20。

表 5-17 实验对象的变异体信息

实验对象	变异体
PBS	AOIS_1, AORB_2, AORB_6, AORB_10, SDL_14, SDL_18, SDL_41, SDL_45, SDL_49
ABBS	AOIS_21, SDL_8, SDL_10, VDL_7
EXP	AOIS_45, ROR_25, SDL_6, VDL_13

5.3.9 潜在的风险分析

影响本文实验结果的一个潜在风险是测试脚本实现的正确性。明显地，不正确的测试脚本直接影响实验结果的正确性。测试脚本经过不同开发人员检查、修正，因此相关测试技术（M-AMT、P-AMT 和 MT）的实验脚本都是正确的。

另一个影响实验结果的风险是度量指标（F-measure、N-measure 和 Time-measure）的可靠性。单次实验结果不能有效地反映度量指标的真实值，本文重复 20 次实验保证度量指标在统计上的可靠性。

5.4 实验结果

5.4.1 M-AMT 和 P-AMT 的故障检测效率

表 5-18 PBS 系统的实验结果

测试技术	F-measure	N-measure
MT	75.60	2.55
M-AMT	55.30	2.65
P-AMT	44.80	4.30

表 5-19 ABBS 系统的实验结果

测试技术	F-measure	N-measure
MT	45.80	3.45
M-AMT	36.10	3.20
P-AMT	33.20	3.60

表 5-20 EXP 系统的实验结果

测试技术	F-measure	N-measure
MT	139.50	2.80
M-AMT	7.70	3.95
P-AMT	108.50	3.0

每一个实验对象的 F-measure 和 N-measure 的均值分别记录在表 5-18~5-20 中。每一个实验对象所有重复实验的 F-measure、N-measure 的结果分布展示在图 5-5~5-10 中。在盒图中，盒子的上边界以及下边界代表一个度量标准的上四分位数和下四分位数，中间的横线表示一个度量标准的中位数，实心圆点表示度量标准的均值，下“胡须”的最小值为下四分位数$-1.5 \times \mathrm{IQR}$，上“胡须”的最大值为上四分位数$+1.5 \times \mathrm{IQR}$，其中IQR表示盒子的长度。不在

两个“胡须”范围内的数值称为异常值，并用“*”表示。

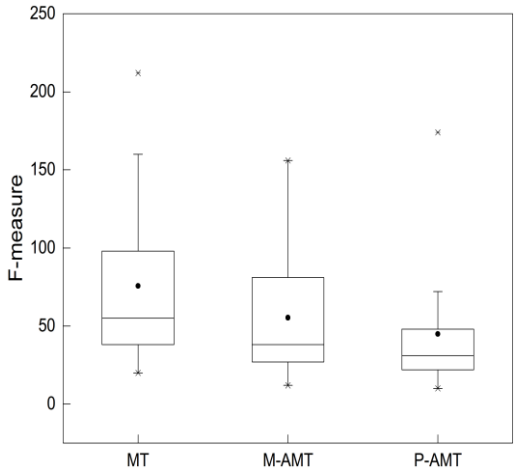


图 5-5 PBS 系统 F-measure 结果

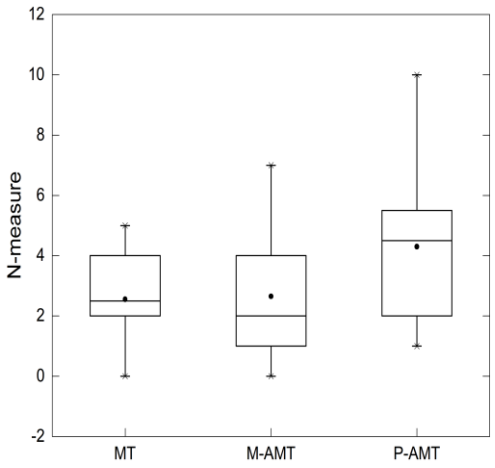


图 5-6 PBS 系统 N-measure 结果

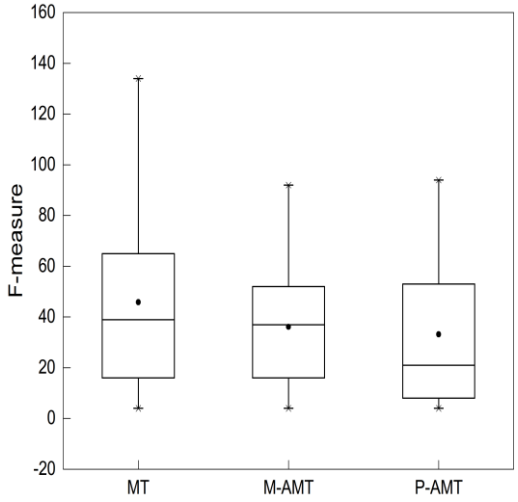


图 5-7 ABBS 系统 F-measure 结果

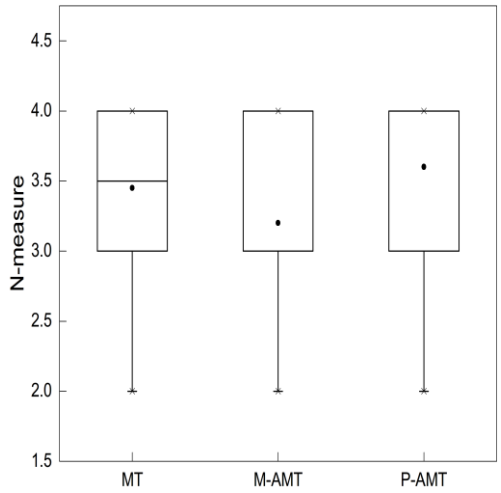


图 5-8 ABBS 系统 N-measure 结果

从测试技术原理、F-measure 和 N-measure 三个方面比较 M-AMT、P-AMT 以及 RT 的故障检测效率。由表 5-18~5-20 和图 5-5~5-10 可知：

（1） 在故障检测效率方面：传统 MT 中测试用例的执行采用随机的方式，即在软件的输入域中随机地选测试用例，该方法实现简单，但是没有利用测试过程信息。M-AMT 和 P-AMT 在 MT 的基础上，引入控制论的思想，通过分析当前测试结果选取更有可能揭示故障的测试用例。实例研究的结果表明：M-AMT 和 P-AMT 比 MT 具有更高的故障检测效率。

（2） 从揭示第一个故障需要的测试用例数目看来，M-AMT 和 P-AMT 的 F-measure 的均值都小于 MT。另外，2/3 的场景中，P-AMT 揭示第一个故障需要的测试用例数目少于 M-AMT（参见图 5-2 和图 5-4）。

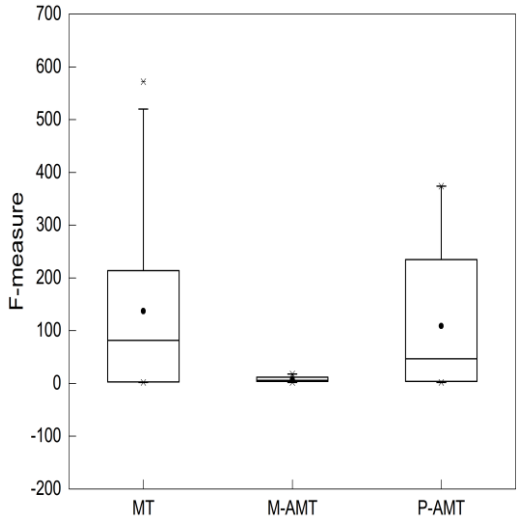


图 5-9 ABBS 系统 F-measure 结果

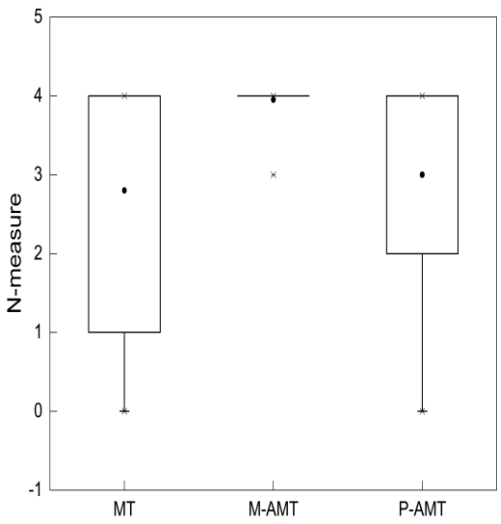


图 5-10 ABBS 系统 N-measure 结果

（3） 从执行一定数目测试用例揭示的故障数目看来，P-AMT 的 N-measure 的均值都高于 MT；M-AMT 的 N-measure 均值在 2/3 的场景中高于 MT（参见图 5-6 和图 5-10）；P-AMT 的 N-measure 均值在 2/3 的场景中高于 M-AMT（图 5-6 和图 5-8）。

综上所述，上述三种测试技术中，P-AMT 的故障检测效率最高，M-AMT 故障检测效率次之，MT 的故障检测效率最低。

5.4.2 时间开销

表 5-21 PBS 系统选择测试用例的时间开销

测试技术	Time-measure（F）	Time-measure（N）
MT	5.76ms	10.24ms
M-AMT	28.45ms	71.70ms
P-AMT	28.95ms	37.05ms

表 5-22 ABBS 系统选择测试用例的时间开销

测试技术	Time-measure（F）	Time-measure（N）
MT	3.79ms	15.24ms
M-AMT	28.85ms	137.00ms
P-AMT	21.90ms	86.25ms

表 5-23 EXP 系统选择测试用例的时间开销

测试技术	Time-measure（F）	Time-measure（N）
MT	29.05ms	4.15ms
M-AMT	4.1ms	51.60ms
P-AMT	23.6ms	62.45ms

表 5-21~5-23 记录了在不同的度量指标下三个实验选择测试用例的时间开销，Time-measure（F）表示在 F-measure 度量指标下选择测试用例花费的时间，Time-measure（N）表示在 N-measure 度量指标下选择测试用例花费的时间。

从测试技术原理、Time-measure 两个方面比较 M-AMT、P-AMT 以及 MT 的时间开销。由表 5-21~5-23 可知：

（1） 在测试技术的性能方面：传统 MT 在软件的输入域中随机地选择测试用例并执行。M-AMT 和 P-AMT 需要根据当前的测试结果做出相应的调整，接着选择测试用例并执行。实例研究的结果表明：MT 选择测试用例的时间短于 M-AMT 和 P-AMT。

（2） 从开始测试到测试结束选择测试用例需要的时间看来，MT 选取测试用例需要的时间在 5/6 的场景中比 M-AMT 和 P-AMT 低。另外 M-AMT 选择测试用例的时间在 3/6 的场景中比 P-AMT 低，即 M-AMT 和 P-AMT 选择测试用例的时间开销相近。

综上所述，MT 选取测试用例的时间开销低于 M-AMT 和 P-AMT，并且 MT、M-AMT 和 P-AMT 选取测试用例的时间开销的均值低于 1 秒。以上情况说明 M-AMT 和 P-AMT 在提升故障检测效率的同时增加了选取测试用例的时间开销，但是增加的时间开销在可以接受的范围内。

5.5 小结

本章使用三个实例程序评估了本文提出的适应性蜕变测试技术 M-AMT 和 P-AMT 的故障检测的有效性与效率。基于实验评估结果，M-AMT 和 P-AMT 的故障检测效率均高于 MT，并且 P-AMT 具有更好的故障检测效率。

6 工作总结与展望

随着软件的规模不断增大、复杂性不断提高，软件测试面临严峻地挑战。当测试预期不存在时，很多测试技术不再能够测试软件。蜕变测试是一种有效缓解测试预期问题的测试技术。在提高蜕变测试的效率方面，研究者把注意力集中在生成和选择“较好”的蜕变关系（测试用例的执行结果容易违反这种蜕变关系）上，忽略了原始测试用例对蜕变测试效率的影响。本文利用适应性分区测试技术，控制蜕变测试的执行过程，提出了两种适应性蜕变测试技术，提高了蜕变测试的故障检测效率，开发了相应的支持工具，具体来说：

(1) **提出两种适应性蜕变测试技术：**本文将适应性分区测试技术中的控制论思想引入到蜕变测试中，提出了两种适应性蜕变测试技术：（a）以蜕变关系为中心的适应性蜕变测试技术，旨在提高蜕变测试的故障检测效率，首先根据原始测试用例与衍生测试用例的测试结果是否违反蜕变关系更新测试剖面，然后依据测试剖面选择分区，最后将选中分区涉及到的蜕变关系作为下一个待验证的蜕变关系；（b）以分区为中心的适应性蜕变测试技术，将蜕变测试中的蜕变关系作为适应性分区测试中测试结果的判断机制，旨在缺少测试预期时解决适应性分区测试技术不能测试的问题，首先根据测试剖面选择分区，然后在选中的分区中随机选择原始测试用例，依据与选中的原始测试用例相关的蜕变关系生成衍生测试用例，最后执行原始测试用例与衍生测试用例，并根据执行的结果是否违反蜕变关系更新测试剖面。

(2) **开发了适应性蜕变测试的支持工具 APT2MT：**该工具支持测试者选择不同的测试技术（M-AMT、P-AMT 和 MT）测试待测程序、设置终止条件、显示分区信息、执行测试和打印测试报告。

(3) **对提出的两种适应性蜕变测试技术进行经验研究：**采用三个真实的 Java 程序实例验证并评估提出技术的有效性与支持工具的实用性。实验表明：在揭示第一个故障方面，M-AMT 和 P-AMT 的故障检测效率高于 MT；在执行一定数目的测试用例方面，M-AMT 和 P-AMT 揭示故障的数目多于 MT。

工作不足及未来展望：

(1) **采用更多的程序进行经验研究：**本文使用了三个真实的程序进行实例验证，研究对象的数量较少，未来需要使用更多的实验对象评估本文提出的两种适应性蜕变测试技术的有效性。

(2) **完善适应性蜕变测试支持工具 APT2MT：**本文开发的适应性蜕变测试支

持工具 APT2MT 缺少识别蜕变关系以及编辑分区规则等功能，未来需要进一步完善 APT2MT 的功能，提高适应性蜕变测试技术的自动化程度。

参考文献

[1] 单锦辉, 姜瑛, 孙萍. 软件测试研究进展 [J]. 北京大学学报(自然科学版), 2005, 41(1): 134-145.

[2] M. Harman, P. McMinn, M. Shahbaz. A Comprehensive Survey of Trends in Oracles for Software Testing, CS-13-01 [R]. Department of Computer Science, University of Sheffield, 2013.

[3] E. T. Barr, M. Harman, P. Minn, M. Shahbaz, S. Yoo. The Oracle Problem in Software Testing: A Survey [J]. IEEE Transaction on Software Engineering, 2014, 41(5): 507-525.

[4] R. M. Hierons. Oracles for Distributed Testing [J]. IEEE Transaction on Software Engineering, 2012, 38(3): 629-641.

[5] T. Y. Chen, S. C. Cheung, S. M. Yiu. Metamorphic Testing: A New Approach for Generating Next Test Cases, HKUST-CS98-01 [R]. Department of Computer Science, Hong Kong University of Science and Technology, 1998.

[6] K.-Y. Cai. Optimal Software Testing and Adaptive Software Testing in the Context of Software Cybernetics [J]. Information and Software Technology, 2002, 44(14): 841–855.

[7] C.-A. Sun, H. Dai, H. Liu, T. Y. Chen, K.-Y. Cai. Adaptive Partition Testing [J]. IEEE Transactions on Computers, (Accepted with Major Revision, 25 May 2018).

[8] S. Segura, G. Fraser, A. B. Sanchez, A. R. Cortes. A survey on metamorphic testing [J]. IEEE Transactions on Software Engineering, 2016, 42(9): 805-824.

[9] R. Wang, K. Ben. Classification of Metamorphic Relations and Its Application [J]. American Journal of Engineer and Technology Research, 2011, 11(12): 1664-1668.

[10] R. Hamlet. Random Testing [M]. Encyclopedia of Software Engineering. John Wiley & Sons, Inc. 2002:970--978.

[11] E. J. Weyuker and B. Jeng. Analyzing Partition Testing Strategies [J]. IEEE Transactions on Software Engineering, 1991, 17(7): 703–711.

[12] D. Hamlet and R. Taylor. Partition Testing Does not Inspire Confidence [J]. IEEE Transactions on Software Engineering, 1990, 16(12): 1402–1411.

[13] T. Y. Chen and Y. T. Yu. On the Relationship Between Partition and Random Testing [J]. IEEE Transactions on Software Engineering, 1994, 20(12): 977–980.

[14] T. Y. Chen and Y. T. Yu. On the Expected Number of Failures Detected by Subdomain Testing and Random Testing [J]. IEEE Transactions on Software Engineering, 1996, 22(2): 109–119.

[15] W. J. Gutjahr. Partition Testing vs. Random Testing: The Influence of Uncertainty [J]. IEEE Transactions on Software Engineering, 1999, 25(5): 661–674.

[16] K.-Y. Cai, T. Jing, and C.-G. Bai. Partition Testing with Dynamic Partitioning [C]. Proceedings of the 29th Annual International Conference on Computer Software and Applications Conference (COMPSACW'05), IEEE Computer Society, 2005, pp. 113–116.

[17] K.-Y. Cai, B. Gu, H. Hu, and Y.-C. Li. Adaptive Software Testing with Fixed-Memory Feedback [J]. Journal of Systems and Software, 2007, 80(8): 1328–1348.

[18] R. A. DeMillo, R. J. Lipton, F. G. Sayward. Hints on Test Data Selection: Help for the Practicing Programmer [J]. IEEE Computer, 1978, 11(4): 34-41.

[19] J. Zhang, J. Chen, D. Hao. Search-Based Inference of Polynomial Metamorphic Relations [C]. Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering (ASE'14), ACM Press, 2014, pp. 701-712.

[20] F. H. Su, J. Bell, C. Murphy. Dynamic Inference of Likely Metamorphic Properties to Support Differential Testing [C]. Proceedings of the 10th International Workshop on Automation of Software Test (AST'15), Co-located with the 37th IEEE International Conference on Software Engineering (ICSE'15), IEEE Computer Society, 2015, pp. 55-59.

[21] C.-A. Sun, Y. Liu, Z. Wang, W.K. Chan. μMT: A Data Mutation Directed Metamorphic Relation Acquisition Methodology [C]. Proceeding of the First International Workshop on Metamorphic Testing (MET'16), Co-located with ICSE 2016, IEEE Computer Society, 2016, pp. 12-18.

[22] A. Gotlieb, B. Botella. Automated Metamorphic Testing [C]. Proceedings of the 27th Annual International Conference on Computer Software and Applications (COMPSAC'03), IEEE Computer Society, 2003, pp. 34-40.

[23] H. Liu, X. Liu, T. Y. Chen. A New Method for Constructing Metamorphic Relations [C]. Proceedings of the 12th International Conference on Quality Software (QSIC'12), IEEE Computer Society, 2013, pp. 59-68.

[24] T. Y. Chen, P. L. Poon, X. Xie. METRIC: Metamorphic Relation Identification Based on the Category-Choice Framework [J]. Journal of Systems and Software, 2016, 116: 177-190.

[25] T. J. Ostrand, M. J. Balcer. The Category-Partition Method for Specifying

and Generating Fuctional Tests [J]. Communications of the Acm, 1988, 31(6): 676-686.

[26] G. Batra, J. Sengupta. An Efficient Metamorphic Testing Technique Using Genetic Algorithm [C]. Proceedings of 5th International Conference on Information Intelligence, Systems, Technology and Management (ICISTM'11), Springer, 2011, pp. 180-188.

[27] G. Dong, T. Guo, P. Zhang. Security Assurance with Program Path Analysis and Metamorphic Testing [C]. Proceedings of the 4th International Conference on Software Engineering and Service Science (ICSESS'13), IEEE Computer Society, 2013, pp. 193-197.

[28] T. Y. Chen, F. C. Kuo, Y. Liu. Metamorphic Testing and Testing with Special Values [C]. Proceedings of the 5th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD'04), 2004, pp. 128-134.

[29] 刘益强. 基于符号执行的蜕变测试中原始测试用例生成技术与支持工具研究 [D]. 北京科技大学, 2016.

[30] X. Xie, W. E. Wong, T. Y. Chen, B. Xu. Metamorphic Slice: An Application in Spectrum-Based Fault Localization [J]. Information and Software Technology, 2013, 55(5): 866-879.

[31] G. Dong, S. Wu, G. Wang, T. Guo, Y. Huang. Security Assurance with Metamorphic Testing and Genetic Algorithm [C]. Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT'10), IEEE Computer Society, 2010, pp. 397-401.

[32] C. Murphy, K. Shen, G. Kaiser. Using JML Runtime Assertion Checking to Automate Metamorphic Testing in Applications Without Test Oracles [C]. Proceedings of the 2nd International Conference on Software Testing Verification and Validation (ICST'09), IEEE Computer Society, 2009, pp. 436-445.

[33] H. Liu, I. I. Yusuf, H. W. Schmidt. Metamorphic Fault Tolerance: An Automated and Systematic Methodology for Fault Tolerance in The Absence of Test Oracle [C]. Proceddings of the 36th International Conference on Software Engineering (ICSE'14), ACM Press, 2014, pp. 420-423.

[34] C.-A. Sun, G. Wang, B. Mu. Metamorphic Testing for Web Services: Framework and A Case Study [C]. Proceedings of the 9th IEEE International Conference on Web Services (ICWS'11), IEEE Computer Society, 2011, pp. 283-290.

[35] C.-A. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, T. Y. Chen. A Metamorphic Relation-Based Approach to Testing Web Services without Oracles [J]. International Journal of Web Services Research, 2012, 9(1): 51-73.

[36] W. K. Chan, S. C. Cheung, K. R. Leung. Towards a Metamorphic Testing Methodology for Service-Oriented Software Applications [C]. Proceedings of 5th International Conference on Quality Software (QSIC'05), IEEE Computer Society, 2005, pp. 470-476.

[37] W. K. Chan, S. C. Cheung, K. R. Leung. A Metamorphic Testing Approach for Online Testing of Service-Oriented Software Applications [J]. International Journal of Web Services Research, 2007, 4(2): 61-72.

[38] J. Mayer, R. Guderlei. On Random Testing of Image Processing Applications [C]. Proceedings of the 6th International Conference on Quality Software (QSIC'06), IEEE Computer Society, 2006, pp. 85-92.

[39] F. C. Kuo, S. Liu, T. Y. Chen. Testing a Binary Space Partitioning Algorithm with Metamorphic Testing [C]. Proceedings of the ACM Symposium on Applied Computing (SAC'11), ACM Press, 2011, pp. 1482-1489.

[40] T. H. Tse, S. S. Yau. Testing Context-Sensitive Middleware-Based Software Applications [C]. Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC'04). IEEE Computer Society, 2004, pp. 458-466.

[41] W. K. Chan, T. Y. Chen, S. C. Cheung, T. H. Tse, Z. Zhang. Towards the Testing of Power-Aware Software Applications for Wireless Sensor Networks [C]. Proceedings of the 12th Ada-Europe International Conference on Reliable Software Technologies (ICRST'07), Springer, 2007, pp. 84-99.

[42] C.-A. Sun, Z. Wang, G. Wang. A Property-based Testing Framework for Encryption Programs [J]. Frontiers of Computer Science, Springer, 2014, 8(3): 478-489.

[43] P. E. Ammann and J. C. Knight. Data Diversity: an Approach to Software Fault Tolerance [J]. IEEE Transactions on Computers, 1988, 37(4): 418–425.

[44] G. B. Finelli. NASA Software Failure Characterization Experiments [J]. Reliability Engineering and System Safety, 1991, 32(1): 155–169.

[45] K.-Y. Cai, H. Hu, and F. Ye. Random Testing with Dynamically Updated Test Profile [C]. Proceedings of the 20th International Symposium on Software Reliability Engineering (ISSRE'09), IEEE Computer Society, 2009, pp. 198.

[46] Y. Li, B. B. Yin, J. Lv, and K.-Y. Cai. Approach for Test Profile Optimization in Dynamic Random Testing [C]. Proceedings of the 39th IEEE Annual

International Computer Software and Applications Conference (COMPSAC'15), IEEE Computer Society, 2015, pp. 466–471.

[47] Z. Yang, B. Yin, J. Lv, K.-Y. Cai, S. S. Yau, and J. Yu. Dynamic Random Testing with Parameter Adjustment [C]. Proceedings of the 6th IEEE International Workshop on Software Test Automationthe, Co-located with the 38th IEEE Annual International Computer Software and Applications Conference (COMPSAC'14), IEEE Computer Society, 2014, pp. 37–42.

[48] J. Lv, H. Hu, and K.-Y. Cai. A Sufficient Condition for Parameters Estimation in Dynamic Random Testing [C]. Proceedings of the 3rd IEEE International Workshop on Software Test Automationthe, Co-located with the 35th IEEE Annual International Computer Software and Applications Conference (COMPSAC'11), IEEE Computer Society, 2011, pp. 19–24.

[49] C.-A. Sun, G. Wang, K.-Y. Cai, T. Y. Chen. Towards Dynamic Random Testing for Web Services [C]. Proceedings of the 36th IEEE International Computer Software and Application Conference (COMPSAC'12), 2012, pp. 164-169.

[50] 张自强. 基于反馈机制的 Web 服务测试技术与工具研究 [D]. 北京科技大学, 2017.

[51] C.-A. Sun, G. Wang, Q. Wen. MT4WS: An Automated Metamorphic Testing System for Web Service [J]. International Journal of High Performance Computing and Networking, 2016, 9(1/2): 104-115.

[52] P. Wu. Iterative metamorphic testing [C]. Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC' 05), IEEE Computer Society, 2005, pp. 19-24.

[53] C.-A. Sun, Y. Liu, Z. Wang, Q. Wen, P. Wu, T. Y. Chen. An Empirical Study on Iterative Metamorphic Testing for Web Services [J]. International Journal on Software Tools for Technology (STTT). (under review)

[54] C.-A. Sun, Y. Liu, Z. Wang. uMT: A Data Mutation Directed Metamprphic Relation Acquisition Methodology [C]. Proceedings of the 1st Internatioal Workshop on Metamorphic Testing (MET'16), ACM Press, 2016, pp. 12-18.

[55] J. Offutt, Y. S. Ma, Y. R. Kwon. An experimental mutation system for Java [J]. ACM Sigsoft Software Engineering Notes, 2004, 29(5):1-4.

作者简历及在学研究成果

一、 作者入学前简历

起止年月	学习或工作单位	备注
2012 年 09 月至 2016 年 06 月	在中国矿业大学（北京）信息与计算科学专业攻读学士学位	

二、 在学期间从事的科研工作

[1] 面向 SOA 软件的蜕变测试技术研究, 国家自然科学基金面上项目(61370061), 主要参与人员.

三、 在学期间所获的科研奖励

[1] 研究生一等学业奖学金, 北京科技大学, 2016.10.

[2] 研究生一等学业奖学金, 北京科技大学, 2017.10.

[3] 优秀三好研究生干部, 北京科技大学, 2017.10.

四、 在学期间发表的论文

[1] 孙昌爱, 代贺鹏，张自强. 面向 Web 服务的动态随机测试系统, 软件著作权, 登记号:2018SR271444, 授权日期:2018 年 4 月 20 日.

[2] 孙昌爱, 代贺鹏，张在兴. 特征模型驱动的适应性服务组装构造支持系统, 软件著作权, 登记号:2018SR278453, 授权日期:2018 年 4 月 24 日.

[3] Chang-ai Sun, Hepeng Dai, Huai Liu, Tsong Yueh Chen, Kai-Yuan Cai. Adaptive Partition Testing. IEEE Transactions on Computers, (Accepted with Major Revision, 25 May 2018).

独创性说明

本人郑重声明：所呈交的论文是我个人在导师指导下进行的研究工作及取得研究成果。尽我所知，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的研究成果，也不包含为获得北京科技大学或其他教育机构的学位或证书所使用过的材料。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做了明确的说明并表示了谢意。

签名：_____ 日期：_____

关于论文使用授权的说明

本人完全了解北京科技大学有关保留、使用学位论文的规定，即：学校有权保留送交论文的复印件，允许论文被查阅和借阅；学校可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。

（保密的论文在解密后应遵循此规定）

签名：_____ 导师签名：_____ 日期：_____

学位论文数据集

关键词*	**密级***	**中图分类号***	**UDC**	**论文资助**
随机测试，适应性分区测试，蜕变测试，测试预期	公开	TP311	004.41	
学位授予单位名称*		**学位授予单位代码***	**学位类别***	**学位级别***
北京科技大学		10008	工学	硕士
论文题名*		**并列题名**		**论文语种***
蜕变测试与适应性分区测试的集成方法与支持工具研究				中文
作者姓名*	代贺鹏		**学号***	G20168664
培养单位名称*		**培养单位代码***	**培养单位地址**	**邮编**
北京科技大学		10008	北京市海淀区学院路 30 号	100083
学科专业*		**研究方向***	**学制***	**学位授予年***
软件工程		软件测试	2 年	2018
论文提交日期*	2018 年 5 月 4 日			
导师姓名*	孙昌爱		**职称***	教授
评阅人	**答辩委员会主席***		**答辩委员会成员**	
电子版论文提交格式 文本（√） 图像（ ） 视频（ ） 音频（ ） 多媒体（ ） 其他（ ） **推荐格式：**application/msword；application/pdf				
电子版论文出版（发布）者		**电子版论文出版（发布）地**		**权限声明**
论文总页数*	57 页			
共 33 项，其中带*为必填数据，为 22 项。				