UNIVERSITY OF SCIENCE AND TECHNOLOGY BEIJING

SOFTWARE TESTING AND SERVICE COMPUTING LAB

TOWARDS DYNAMIC RANDOM TESTING FOR WEB SERVICES

technical report

姓名 : 代贺鹏

版本 : 1

2018 年 1 月 26 日

目录

1 问题阐述

面向服务的架构(Service Oriented Architecture, SOA)提出了一种软件开发的泛型 [21]。Web 服务是 SOA 软件的基本组成单元， 它们将接口发布到一个恰当的仓库中实现暴露功能的目的。发布的接口可以被其它的 Web 服务或者用户搜索、使用。当某些 Web 服务实现了重要的业务逻辑时， 确保它们的质量成为一个严峻的问题。

软件测试是可以保障 Web 服务质量的一种系统的、可行的方法。然而由于 Web 服务的自身特点导致测试面临诸多挑战 [5]。 并且发布的 Web 服务一般都是经过测试的，因此服务中的故障不容易被揭示。随机测试(Random Testing,RT) [13] 是一种被广泛采用的软件测试技术。RT 在待测程序的输入域中随机、独立地挑选测试用例，因此 RT 策略容易使用。然而 RT 策略没用利用待测程序本身的信息以及测试过程信息，导致一些情况下测试效率不高。当软件中的故障容易揭示时，简单易用的 RT 策略是较好的选择。但是由于已经发布的 Web 服务故障不容易揭示，RT 策略可能不能充分保障 Web 服务的质量。动态随机测试(Dynamic Random Testing,DRT) 在 RT 策略的基础上增加了软件的控制理论 [2]，许多研究 [16, 18, 25]表明 DRT 可以提高 RT 的测试效率。利用 DRT 比 RT 具有更高的故障检测效率这一特点，本文提出了面向 Web 服务的动态随机测试策略。主要的研究工作如下：

- 提出一种测试基于 SOA 的 Web 服务的技术。我们提出了将 DRT 策略运用到 Web 服务的测试框架以及半自动测试工具。
- 利用三个真实存在的 Web 服务验证我们提出的 DRT 测试框架的合理性。
- 探究分区以及参数对 DRT 策略效率的影响。

接下来的技术报告分为如下几个部分：第二章，介绍研究相关的概念与技术，包括：面向服务的架构、Web 服务的测试、变异测试、分区测试以及动态随机测试； 第三章，介绍将动态随机测试技术测试 Web 服务的架构；第四章，介绍动态随机测试技术测试 Web 服务的半自动化测试工具；第五章，介绍实验对象的基本情况以及实验设置并且 展示实验结果；第六章，总结全文。

2 相关理论

本章介绍研究相关的概念与技术，包括面向服务的架构、Web 服务的测试、随机测试与分区测试、动态随机测试、变异分析。

2.1 面向服务的架构

面向服务的架构(Service-Oriented Architecture,SOA)是一种软件工程方法学或者软件开发范型。该架构定义了 Internet 环境下松散耦合的、基于标准的、面向服务的应用程序 开发模式。SOA 的一般模型 [21]如图 1 所示。

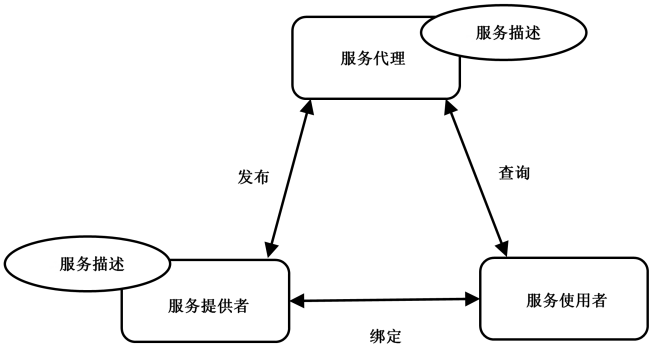


图 1: SOA一般模型

该模型中存在三种角色：服务提供者、服务使用者和服务代理。服务提供者对服务的功能使用操作接口方式描述，并通过 WSDL(Web Services Description Language)和 UDDI(Universal Description Discovery and Integration) [22]将服务发布到服务代理，以便服务使用者发现、访问该服务。服务发布后，服务使用者 通过查询服务代理，绑定查询结果对应的服务，根据服务描述的操作接口调用服务功能。

SOA 软件开发包括服务与服务组合两个层次的设计与实现。现有的 SOA 软件开发模型借鉴了构件化软件开发模型的思想，将服务看作一种构建，将服务组合看作构建组合。与传统 软件开发方法学相比，SOA 软件开发的特点在于：

- 服务之间耦合更加松散。“高内聚、低耦合”试软件设计的一般原则，与函数、过程、类或者构建等传统应用程序开发的基本单元相比，服务之间的饿耦合度更低。
- 服务的定义与实现进一步分离。对象与构建都支持定义与实现分离，但服务的定义与实现分离更加彻底：在不同的文件中分别定义与实现服务，同一服务可以由不同的 编程语言实现并且服务的逻辑实现与物理是实现也可以是分离的。

- 服务可以发布、注册与使用。服务的开发与使用比任何传统软件资产更加独立和开放，从而解决了传统软件开发方法学中的互操作问题。

SOA 中服务通常实现一个软件系统的一部分，通过对外提供一组操作参与分布式计算，是一个大粒度的单元。服务由描述和实现两部分组成。服务描述进一步分为基本描述部分和 附加描述部分。基本描述部分由服务开发者提供，着重描述服务的规格说明，包括：服务接口、异常、操作语义、服务质量、合约条件以及测试信息等。附加描述部分指明服务的 分级，度量的服务的质量，可信性使用历史等。

2.2 Web服务的测试

Web 服务是 SOA 软件的基本组成单元，确保服务描述与实现的一致性是 SOA 软件质量保证的重要方面。软件测试是保证软件质量的重要方法。与传统软件开发模型相比，SOA 除了改变系统的构建和使用方式外，同样给测试带来改变。在 SOA 环境下，服务不是在使用者的系统上物理地集成，而是通过调用地方式，将服务提供的功能组合为所需要的业务逻辑。SOA 的上述特点增加了测试的复杂性 [6]：

- 基于服务的系统分散的分布，须在不同的配置下保证服务的质量。
- 由于服务的松散耦合，系统中的服务相互独立地变化，对迭代测试产生影响。
- 服务的定义与实现分离，同一个服务描述在不同时间可能使用不同的实验方式。系统的功能随需求变化，表现为添加或替换某些服务，增加了集成测试的困难。
- 服务描述由服务提供者给出，服务代理和服务使用者难以判断可信度、完整度，使得设计测试用例变得困难。
- 服务由多个不同的组织或机构共有，难以协调它们共同完成测试

SOA 环境下服务独特的使用方式使其测试面临若干挑战 [6]：

- **缺少服务源代码：** 对于服务代理和服务使用者，服务仅仅是一系列操作接口，源代码不可见。因此传统的百合测试方法难以使用。
- **运行的动态性：** 传统的软件架构在测试时通常能够确定待测组件或者一个可能的组件集合，例如面向对象系统的多态性。然而 SOA 工作流程中调用究竟哪个服务难以确定。换言之，SOA 中服务运行的动态性较强，难以预料使用场景。

- **缺乏控制：** 传统软件测试时，测试者拥有对待测程序的实际控制。然而在 SOA 中，近服务提供者对服务拥有控制权。服务代理和服务使用者无法掌握服务的变化情况。

- **可信度未知：** 服务提供者提供的服务描述在实际中可能是不正确的，提供错误描述的服务给测试带来了不确定性。

由此可以看出，SOA 环境下的服务测试相较于传统软件更加困难。随机测试在软件测试中占有大量的比重，但是根据随机测试本身的特点在一些情况下效率不高，因此找高一种更加高效的测试方法是一件重要的事情。

2.3 随机测试和分区测试

随机测试(Random Testing,RT) [13]和分区测试(Partition Testing,PT) [24]是两个主要的软件测试方法。随机测试在选取测试用例时一般按照均等的概率或者根据功能剖面在输入域中随机选取测试用例。所谓功能剖面，是根据待测软件不同输入数据的使用频率人为设定概率： 使用频率高的输入数据对应的测试用例则有较高的选取概率，使用频率低的输入数据赋予较低的选择概率。随机测试选取测试用例的方法较为简单，有助于估计待测软件的稳定性与可信性。

与 RT 策略不同，PT 策略旨在同构的产生测试用例以提高测试效率。PT 策略首先将输入域划分成若干个不相交的子分区，然后从每一个分区中挑选测试用例进行测试。在 PT 策略中，分区期望存在同构的性质，即相同分区下的测试用例应该使程序产生相似的行为。

从 1980 年甚至更早时期以来，RT 和 PT 在测试效率方面进行了大量的对比 [12] [7] [9] [11]。 虽然 PT 策略更系统地产生测试用例，但是 PT 策略没有大幅度改进 RT 策略，在某些情况下甚至不如 RT 策略。PT 策略在理论上希望分区内地测试用例具有同构地性质，然而在现实世界中同构地性质很难保证。因此在一些情况下 PT 策略的效率可能较低。

为了进一步提高分区测试的测试效率，Cai [4] [3]将 RT 策略与 PT 策略结合提出一种新的测试技术：随机分区测试(Random Partition Testing,RPT)。在 RPT 策略中，输入域首先被划分为 m 个不相交的分区 $c_1, c_2, \ldots, c_m$，每一个分区 c_i 被指派一个被选择的概率 p_i 。然后根据测试剖面 $\{p_1, p_2, \ldots, p_m\}$ 随机选择分区，其中 $p_1 + p_2 + \cdots + p_m = 1$。最后再从选中的分区 c_i 中随机选择测试用例。

2.4 动态随机测试

在随机分区测试策略中，一个分区对应的选择概率在整个的测试过程中是不变的，这一点可能不总是好的。因为引起故障的输入在输入域中趋向于聚簇在连续的区域 [15] [1] [10]，也就是说存在一些分区更可能揭示软件中的故

障。Cai 等人依据这一想法，利用软件的控制理论 [2]提出了动态随机测试策略（DRT) [4]以改进传统的随机测试与随机分区测试策略。软件的控制理论探索软件工程理论与控制理论相互作用的关系，被用来解决软件工程中的问题。DRT 策略的主要特点是在测试的过程中根据每一次测试用例的执行结果动态改变测试剖面：假设存在一个分区 c_i ，若该分区中的一个测试用例 t 揭示了软件中的故障，那么认为该分区具有较高的故障检测能力，因此增大该分区被选择的概率，即 $p_i + \varepsilon$。如果这个测试用例没有检测出故障，则减小该分区被选择的概率，即 $p_i - \varepsilon$。DRT 策略调整测试剖面的具体步骤如下：

将待测软件的输入域划分为 m 个不相交的分区：$c_1, c_2, \ldots, c_m$，每一个分区中有 $k_1, k_2, \ldots, k_m$ 个测试用例。初始化参数 ε，并且 $\varepsilon > 0$。 根据每一个分区所对应的概率 p_i 随机选取一个分区 c_i,在这里 $p_1 + p_2 + \cdots + p_m = 1$。随机地从 c_i 中挑选一个测试用例 t。 如果该测试用例 t 揭示了软件中的故障，增大测试用例 t 所在分区被选择的概率 p_i，同时减小其它分区被选择的概率 $p_j (j \neq i)$，并把缺陷移除。

$$p'_j = \begin{cases} p_j - \dfrac{\varepsilon}{m-1} & \text{if } p_j \geq \dfrac{\varepsilon}{m-1} \\ 0 & \text{if } p_j < \dfrac{\varepsilon}{m-1} \end{cases}, \tag{1}$$

并且

$$p'_i = 1 - \sum_{\substack{j=1 \\ j \neq i}}^{m} p'_j. \tag{2}$$

当测试用例 t 没有揭示故障时，减小测试用例 t 所在分区被选择的概率 p_i，同时增大其它分区被选择的概率 $p_j (j \neq i)$，并把缺陷移除。

$$p'_i = \begin{cases} p_i - \varepsilon & \text{if } p_i \geq \varepsilon \\ 0 & \text{if } p_i < \varepsilon \end{cases} \tag{3}$$

对$\forall j = 1, 2, 3, \ldots, m$ 并且$j \neq i$，有

$$p'_j = \begin{cases} p_j + \dfrac{\varepsilon}{m-1} & \text{if } p_i \geq \varepsilon \\ p_j + \dfrac{p_i}{m-1} & \text{if } p_i < \varepsilon \end{cases} \tag{4}$$

DRT 策略的详细信息展示在算法 2 中。DRT 策略首先根据测试剖面 $\{p_1, p_2, \ldots, p_m\}$ 随机地选择一个分区 c_i(算法1中的第二行)，然后在被选中的分区 c_i 中随机挑选测试用例 t (算法1中的第三行)。如果测试用例 t 揭示了软件中的故障，则增大分区 c_i 被选择的概率，同时减小其它分区 $c_j (j = 1, 2, \ldots, m, j \neq i)$被选择的概率(算法1中的第六行)。如果测试用例 t 没有揭示软件中的故障，则减少分区 c_i 被选择的概率，同时增大其它分区 $c_j (j = 1, 2, \ldots, m, j \neq i)$被选

择的概率(算法1中的第八行)。由此可以看出 DRT 策略最终使得测试剖面符合每一个分区的失效率：失效率大的分区杯选择的概率大，是效率低的分区被选择的概率小。

对于 DRT 策略我们不妨设在一次软件测试过程中将软件的输入域分为 m 个分区。在理想情况下，一个测试用例就可以找出所有的缺陷，这时算法的语句频度为 $T = m + 1$。 由于在实际的情况中分区的个数总是有限的，因此这时算法的渐进时间复杂的为 $O(1)$ 。如果不能一个测试用例就找出所有的缺陷，那么不妨设在整个测试的过程中共执行了 n 个测试用例，这时的语句频度为 $T(n) = h * n + h$。当 n 很大时，这时的时间复杂度为 $O(n)$。

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 profile $\{p_1, p_2, \ldots, p_m\}$
3: Select a test case t from c_i
4: Test the software using t
5: **if** a fault is revealed by t
6: Update p_j $(j = 1, 2, \ldots, m$ and $j \neq i)$ and p_i according to Formulas 1 and 2.
7: **else**
8: Update p_j $(j = 1, 2, \ldots, m$ and $j \neq i)$ and p_i according to Formulas 3 and 4.
9: **end_if**
10: **end_while**

2.5 变异分析

变异分析是一种基于故障的软件测试技术。变异分析的基本思想是针对某一个原始程序，运用变异算子模拟程序中的常见错误并植入源程序，这个过程称为变异生成。生成后的原始程序的错误版本成为变异体。使用若干测试用例分别在变异体以及原始程序上执行，若存在某个测试用例在变异体以及源程序上的执行结果不同，则变异体“被杀死”，即植入的故障被检测到。反之，变异体未被杀死。有些变异体虽然在语法上与原始程序不同，但是在语义上是一致的，因此没有一个测试用例能够杀死变异体，这类变异体称为等价变异体。使用某种测试技术 S 产生原始程序 P 的测试用例集 TS，可以通过原始程序及各个变异体在 TS 上的运行情况对测试技术 S 以及测试用例集 TS 进行评估。上述过程称为变异分析。变异分析中最重要的评估指标称为变异得分(Mutation Score,MS)，它是指测试用例集 TS 中能够杀死的变异体数量占变异体总量的比

例。变异得分的定义如下：

$$MS(P,TS) = \frac{N_k}{N_m - N_e} \tag{5}$$

在公式(5)中，P 是源程序，TS 是某种技术产生的测试用例集。N_k 表示被杀死的变异体数量，N_m 表示变异体的总数量，N_e 是等价变异体的数量。

3 DRT在Web服务中的应用

我们是第一个 [23]提出了 DRT 策略在 Web 服务中的测试框架，并且描述了一个支持该框架的部分自动化原型。

3.1 测试框架

考虑 DRT 策略的原则以及 Web 服务的特点，我们提出了将 DRT 策略运用到 Web 服务的测试框架，如图 2 所示。从图中可以看出 该框架分为上下两

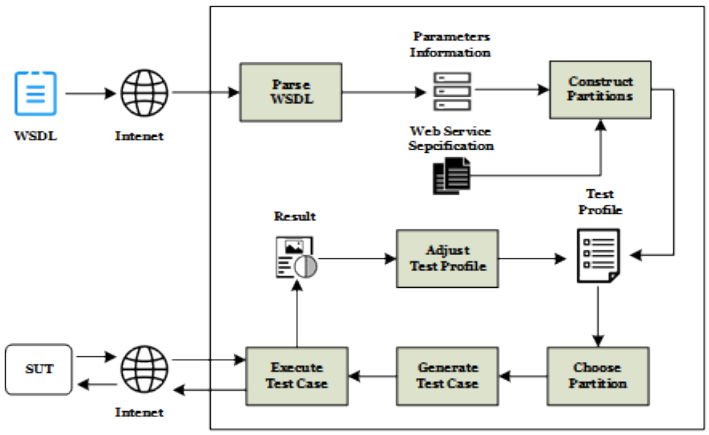


图 2: DRT 运用到 Web 服务的测试框架

个部分：上部为待测的 Web 服务，下部是 DRT 策略的核心步骤。两者的相互作用展示在框架图中。其细节 如下所示：

1. **Partition**。分区测试涉及到许多测试技术 [24]。许多工作 [24] [7]研究了怎么高效、恰当地进行分区。本文将这些技术运用到DRT策略中，因此我们不对分区进行过多讨论。

2. **Test Case Generator**。WSDL 文件定义了服务中各种操作的参数以及数据类型和取值范围。因此可以通过解析 WSDL 文件获取每一个分区涉及到的参数信息，从而为每一个分区产生指定大小的测试用例集。

3. **Probability Distribution Adjustment**。这个组件需要设置、调整测试剖面。设置初始测试剖面的方式有两种：一，采用均等概率分布作为初始测试剖面；二，根据测试人员的经验认为某些分区具有较高的失效率或者这些分区十分重要则分配较大的被选择概率。相应地，其它分区分配较小的被选择概率。

4. **Executor**。这个组件一方面接收来自 Test Case Generator 组件产生的测试用例集，并根据 Probability Distribution Adjustment 组件调整后的

测试剖面选择分区，然后在被选中的分区中挑选测试用例。另一方面，该组件将选中的测试用例转化为输入信息通过 SOAP 协议传给待测服务，并且负责拦截待测服务的返回结果。

5. **Evaluator**。该组件通过比较每一个测试用例的执行结果与该测试用例的预期输出判断此次测试成功还是失败。然后该组件将判断结果传递给 Probability Distribution Adjustment 组件。Probability Distribution Adjustment 组件再根据测试结果调整测试剖面。

4 原型

为了方便地运用DRT技术测试 Web 服务，我们设计、实现了一个部分自动化原型如图 3 所示。首先，测试人员需要输入待测服务的地址并点击 *Parse* 按钮。此时原型将自动分析输入的参数及类型和输出的数据类型。然后从显示的操作列表中选中一个操作。对于分区以及测试用例集，该原型提供两种生成方案：一，自动划分分区和自动产生测试用例集；二，上传事先定义好的分区以及测试用例集。如果选择第一种方法初始测试剖面将自动产生，否则需要测试人员设置初始测试剖面。测试人员设置执行的测试用例上限后，按 Test 按钮开始测试。在测试期间，如果检测出一个故障，该原型将暂时停止测试。这时测试人员可以选择移除故障继续测试或者停止测试。测试结束后该原型将产生一个测试报告供测试人员分析结果。

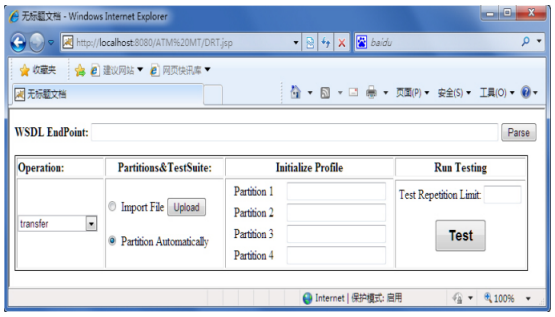


图 3: 原型接口

5 经验学习

我们设计了一系列的经验学习来评估 DRT 策略对 Web 服务的测试效率。这一部分详细地介绍了实验的细节。

5.1 调查的问题

我们的实验中着重调查一下几个问题：

RQ1 DRT 策略对 Web 服务的故障检测效率。

故障检测效率是评价一个测试技术的重要指标。本文探究 DRT 策略在三个真实 Web 服务的故障检测效率。

RQ2 分区多少对 DRT 测试效率的影响。

我们对每一个实验对象设计了两种分区方案：一种是未简化的决策表；另一种是简化之后的决策表。本文探究相同分区方式下，不同分区数目对DRT 策略效率的影响。

RQ3 ε 的取值对DRT测试效率的影响。

我们对 DRT 策略中的参数 ε 的值设置了一系列的梯度，探究参数值的变化对 DRT 策略的故障检测能力的影响。

RQ4 DRT 策略在对 Web 服务进行测试的时间开销。

本文在2.4章节讨论了 DRT 策略需要线性的时间产生测试用例。我们希望通过经验研究的方式评估 DRT 策略在对 Web 服务进行测试时选择测试用例的时间开销。

5.2 实验对象

在我们的研究中，我们对三个真实的 Web 服务的规格说明 [19] [20] [8]进行了实现，分别为：航空行李托运管理服务、联通计费服务、停车计费服务。所有实验对象的详细说明记录在表 1中。每一个实验对象的变异体在RT策略下均需要至少20 个测试用例才能揭示，具体变异情况如表 1的最后一列所示。

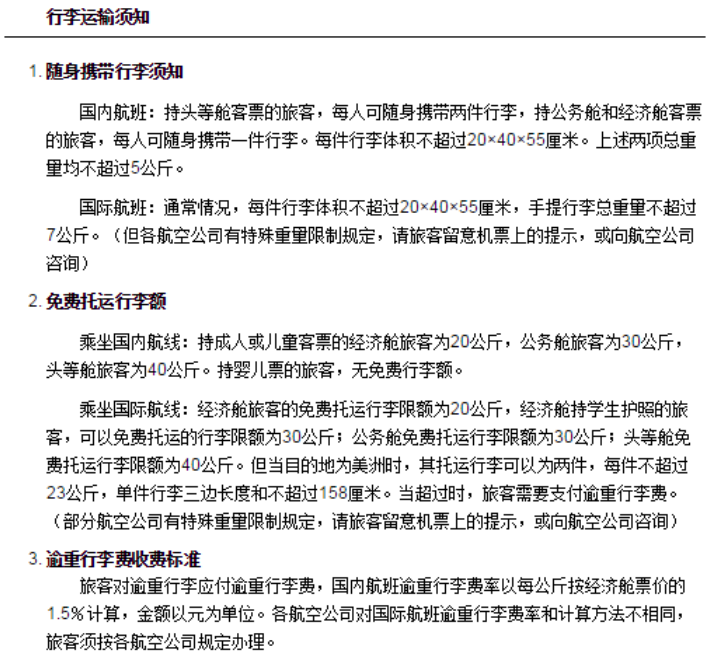
表 1: 实验对象

实验对象	代码行数	变异体数目
航空行李托运	116	2
联通计费	131	11
停车计费	129	4

每一个实验对象的详细情况如下。

5.2.1 航空行李托运

航空行李托运管理服务的实现参照了北京首都国际机场关于行李运输的相关规定，如图 3。该服务实现了可随身携带行李数量查询、行李重量查询、可免费托运的行李的重量查询以及行李计费等功能。



行李运输须知

1. **随身携带行李须知**

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。每件行李体积不超过20×40×55厘米。上述两项总重量均不超过5公斤。

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，请旅客留意机票上的提示，或向航空公司咨询）

2. **免费托运行李额**

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。

乘坐国际航线：经济舱旅客的免费托运行李限额为20公斤，经济舱持学生护照的旅客，可以免费托运的行李限额为30公斤；公务舱免费托运行李限额为30公斤；头等舱免费托运行李限额为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，单件行李三边长度和不超过158厘米。当超过时，旅客需要支付逾重行李费。（部分航空公司有特殊重量限制规定，请旅客留意机票上的提示，或向航空公司咨询）

3. **逾重行李费收费标准**

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 4: 北京首都国际机场行李运输通知

根据飞行的目的地，航班分为国际航班和国内航班。每一架飞机提供三个座舱等级供乘客选择：经济舱、公务舱以及头等舱。不同座舱的乘客可免费托运的行李重量不同，具体的托运计费标准如表 2所示。该表中 $price_0$ 表示经济舱的票价，表中第四行第七列国际航班经济舱免费托运的行李重量根据乘客身份的不同有所差异，当乘客为学生时免费托运的重量为 30kg，否则免费托运的重量为 20kg。

表 2: 航空行李托运计费规则

	国内航班			国际航班		
	头等舱	公务舱	经济舱	头等舱	公务舱	经济舱
随身携带(单位：kg)	5	5	5	7	7	7
免费托运(单位：kg)	40	30	20	40	30	20/30
超重计费(单位：kg)	$price_0 * 1.5\%$			$price_0 * 1.5\%$		

航空行李托运管理服务的输入接口用五元组(at, cc, ecf, pc, bw)表示。其

中 at 是航班类型(Airline Type)的简称，有 2 种取值。当 $at = 0$ 时，表示国内航班；当 $at = 1$ 时，表示国际航班。cc 是座舱等级(Cabin Class) 的简称，有3种取值。当 $cc = 0$ 时表示经济舱；$cc = 1$ 时表示公务舱；$cc = 2$ 时表示头等舱。ecf 是经济舱票价的简称(Economy Class Fare),单位：元。pc 是乘客类型(Passenger Category)的简称，有 4 种取值。当 $pc = 0$ 时，表示婴儿；$pc = 1$ 时表示儿童；$pc = 2$ 时表示学生；$pc = 3$ 时表示成人。bw 是乘客所有行李总重量的简称(Baggage Weight)，单位：kg。该服务的输出为行李的托运费用(Consignment Fare),单位：RMB。

5.2.2 联通计费服务

联通计费服务是根据联通 3G 网络的三种基本套餐进行手机业务计费的一款 Web 服务.该服务提供了一个操作:查询账单(billCaculation)。根据客户选取的套餐类型、月租费用以及实际的语音通话时长和上网流量计算当月用户需要交纳的话费金额。

联通 3G 网络一共提供了 3 种套餐：A, B, C。联通计费服务的输入接口用五元组($pricePackage, monthlyRent, voiceCall, videoCall, flow$)表示。$pricePackage$ 表示套餐类型，并且 $pricePackage \in \{A, B, C\}$，其中 A, B, C 分别代表三种套餐。$monthlyRent$表示月租费用：每种套餐都有多个月租可供选择，具体套餐的月租规格如表 3、 4、 5 所示。$voiceCall$ 表示实际的通话时间 $voiceCall \geq 0$ 并且为整数，单位：min。$flow$ 表示实际的上网流量 $flow > 0$,单位：MB。输出为当月的话费用 bill 表示，单位：￥。

表 3: 联通 3G 套餐 A 细则

套餐月租(￥)		46	66	96	126	156	186	226	286	386	586	886
套餐包含	免费语音(min)	50	50	240	320	420	510	700	900	1250	1950	3000
	免费流量(MB)	150	300	300	400	500	650	750	950	1300	2000	3000
	接听免费	国内(含可视电话)										
超出收费	语音收费(￥/min)	0.25	0.20	0.15								
	流量收费(￥/KB)	0.0003										
	可视电话收费(￥/min)	0.60										

表 4: 联通 3G 套餐 B 细则

套餐月租(￥)		46	66	96	126	156	186
套餐包含	免费语音(min)	120	200	450	680	920	1180
	免费流量(MB)	40	60	80	100	120	150
	接听免费	国内(含可视电话)					
超出收费	语音收费(￥/min)	0.25	0.20	0.15			
	流量收费(￥/KB)	0.0003					
	可视电话收费(￥/min)	0.60					

表 5: 联通 3G 套餐 C 细则

套餐月租(￥)		46	66	96
套餐包含	免费语音(min)	260	380	550
	免费流量(MB)	40	60	80
	接听免费	国内(含可视电话)		
超出收费	语音收费(￥/min)	0.25	0.20	0.15
	流量收费(￥/KB)	0.0003		
	可视电话收费(￥/min)	0.60		

表 6: 停车费率

实际停车时间	停车费率					
	工作日			周末		
	摩托车	两门跑车	轿车	摩托车	两门跑车	轿车
$(0.0, 2.0]$	\$4.00	\$4.50	\$5.00	\$5.00	\$6.00	\$7.00
$(2.0, 4.0]$	\$5.00	\$5.50	\$6.00	\$6.50	\$7.50	\$8.50
$(4.0, 24.0]$	\$6.00	\$6.50	\$7.00	\$8.00	\$9.00	\$10.00

5.2.3 停车计费服务

停车计费系统是一款计算车辆停放费用的 Web 服务。该服务根据车辆的信息和停车时间以及折扣信息进行费用计算。车辆信息包括车的类型：摩托车或汽车。其中汽车又可以分为：跑车或轿车。停车时间分为：工作日或周末。折扣信息包含三个方面：一，客户提供折扣券，则可享受 50% 的优惠；二，客户可以在停车时预计停车时间。系统为客户提供三种预估停车时间：不超过 2 小时、2-4 小时和 4-24 小时。若客户实际停车时间在预估停车时间范围内就可以享受 40% 的优惠。否则要在停车费用的基础上额外承担 20% 的管理费；三，不可以在提供折扣券的同时预估停车时间，但是当两者都没有时，则不享受任何优惠。

系统规定车辆停放不得超过当天 24 点并且根据停车时间采用阶梯式计费方式如 6。

该服务的输入接口用六元组 $(typeOfVehicle, typeOfCar, dayOfWeek, actualParkingDuration, discountCoupon, estimation)$表示。每一个参数的具体含义、数据类型以及合法输入见表 7。该服务的输出为 fee ，表示需要缴纳的停车费($fee > 0$,单位：\$)。

表 7: 停车计费服务输入信息

输入参数	参数含义	数据类型	合法输入	说明
typeOfVehicle	车辆类型	整形	0 或 1	0表示摩托车 1表示汽车
typeOfCar	汽车类型	整形	0 或 1	0表示跑车 1表示轿车
dayOfWeek	停车日	整形	0 或 1	0表示工作日 1表示周末
actualParking Duration	实际停车时间	浮点型	(0.0, 24.0]	停放时间不能超过当天24点
discountCoupon	折扣券	布尔型	true 或 false	true表示提供折扣券 false表示不提供
estimation	预计停车时间	字符串	“(0,2]”, “(2,4]” “(4,24]”,null	折扣券值为true时该值只能为null

5.3 变量

5.3.1 自变量

测试技术是我们研究的自变量。DRT 是其中一个变量，另外 RPT 在 RT 的基础上增加了分区的概念，而 DRT 又在分区的基础上增加了控制理论。因此本文选择 RPT 和 RT 作为 DRT 策略的比较对象。

5.3.2 因变量

因变量用来度量 DRT 策略在 Web 服务中测试效率。有很多度量标准可以判断测试技术是否有效：P-measure (在当前测试用例集中至少检测出一个故障的概率)、E-measure (在当前测试用例集中期望检测的故障数目)、F-measure (当前测试用例集中检测一个故障需要的测试用例数目)、T-measure (当前测试用例集中检测出所有故障需要的测试用例数目)。在这些度量标准中我们认为 F-measure 以及 T-measure 是最合适评估 DRT 测试有效性的度量标准。

在算法 1中当揭示软件中的第一个故障之后测试过程可能并没有终止并且 DRT 策略会根据测试的执行结果调整测试剖面。因此探究第一个故障揭示之后不同测试策略的测试能力是一个很有意思的问题。本文由此提出了一个新的度量指标 NF-measure:当揭示第一个故障之后还需要多少测试用例才能揭示第二个故障。为了表示方便下文分别用F, NF, T代表 F-measure、NF-measure 以及 T-measure。

对于 RQ4,测试时间是恰当的是度量标准。在本文中我们统计每一种测试策略揭示软件中所有故障所需要的平均时间。

5.4 实验设置

5.4.1 分区设置

三个实验对象的分区情况如 8所示。

表 8: 每一个实验对象的分区情况

Web 服务	分区方式1	分区方式2
航空行李托运	24	7
联通计费	20	3
停车计费	18	3

航空托运服务有两种分区方式：一，根据规格说明书将所有的输入进行组合形成原始决策表，决策表中的每一条规则对应一个分区；二，在原始决策表的基础上将具有相同动作的规则进行合并，合并后的决策表每一个规则对应一个分区。

联通计费有两种分区方式：一，根据规格说明书将所有的输入进行组合形成决策表，决策表中的每一条规则对应一个分区；二，仅考虑一个参数：套餐类型A、B、C，按照套餐类型行进分区。

停车计费有两种分区方式：一，根据规格说明书将所有的输入进行组合形成决策表，决策表中的每一条规则对应一个分区；二，仅考虑一个参数：交通工具的类型：摩托车、2 门的汽车和4门的汽车。

5.4.2 测试剖面

由于测试用例是在测试的过程中随机产生的，因此用均等的概率分布作为初始测试剖面是一个保守、可行的方案。另外测试人员也可以根据以往的测试经验将某一个或某些分区被选择的概率增大，并且减少某些分区被选择的概率也是可行的初始剖面的设置方案。

5.4.3 参数设置

在实验中我们对 DRT 策略中的参数 ε 设置了一系列的值 $\varepsilon \in \{0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5\}$ 。$\varepsilon = 0.5$已经是一个很大的数很容易造成这样的情况：当某一个分区内的测试用例没有揭示软件中的故障时，很有可能导致该分区被选择的概率变为0。此时明显是不合理的。例如在停车计费服务中当在分区方式 2 下进行测试时，假如 $\varepsilon = 0.75$ 并且初始测试剖面为均等概率分布，即 $p_i = 1/3$，第一次测试在第一个分区选中一个测试用例，执行后没有揭示软件中的故障。由于 $p_i < \varepsilon/(3-1)$，所以 $p_1 = 0$。即便是失效率很高的分区也不可能每一个测试用例都能揭示软件中的故障，但是当 ε 取值很大时，往往会因为一次的测试结果

将某一个分区被选择的概率直接降为 0，这显然是不合理的。因此 ε 取值($\varepsilon \leq 1$ 始终成立)不宜太大。

5.5 实验环境

本文的实验运行在虚拟机上的 Ubuntu 16.04 64 位操作系统上。在该系统中有 2 个 CPU 和 2GB 的内存。用 Java 语言生成的实验脚本。在实验过程中重复 30 次保证 F-measure、NF-measure 和 T-measure 度量标准的均值可靠。

5.6 有效性分析

- **内部有效性**。内部的效性主要与测试技术的实现有关。实现测试技术需要编写实验脚本。本文的测试代码经过不同的开发人员检查、修正后，我们确信所有的测试技术都正确实现了。

- **外部有效性**。用三个 Web 服务作为实验对象是影响外部的有效性的明显因素。但是本文的三个 Web 服务是根据真实的规格说明实现的。此外本文挑选 17 个至少需要 20 个测试用例才能揭示变异体评价不同测试策略的优劣。即便本文针对 DRT 的参数设置了 13 个不同梯度的数值并且对每一个 Web 服务采取了两种分区模式，我们依然很难保证在其它类型的 Web 服务中得到类似的结果。

- **概念有效性**。本文涉及的三个度量标准，在概念方面容易理解，在运用方面也很简单。因此我们对概念的有效性很有自信。

- **结论有效性**。本文重复了足够的实验次数保证 F-measure、NF-measure 和 T-measure 在统计上的可靠性。

表 9: 航空行李托运服务测试结果

Strategy		Partition Schema 1						Partition Schema 2					
		F	SD_F	NF	SD_{NF}	T	SD_T	F	SD_F	NF	SD_{NF}	T	SD_T
RT		13.30	10.34	28.50	23.95	41.80	27.13	13.30	10.34	28.50	23.95	41.80	27.13
RPT		11.93	11.24	24.36	22.56	35.99	25.02	8.59	8.02	22.47	19.97	27.06	18.73
DRT	1.0E-5	12.31	11.89	24.47	24.20	36.78	26.64	5.80	6.25	20.27	17.93	26.08	19.03
	5.0E-5	11.96	11.62	22.52	21.72	34.48	24.69	6.00	6.90	21.53	19.18	27.54	20.15
	1.0E-4	11.76	11.95	23.84	23.24	35.60	26.03	6.48	7.84	20.54	18.87	27.01	20.04
	5.0E-4	11.46	10.58	22.73	22.21	34.19	23.64	6.21	7.26	21.45	19.66	27.66	20.32
	0.001	12.02	11.60	22.19	21.88	34.21	24.42	6.32	7.46	21.10	19.69	27.43	20.80
	0.005	11.00	9.82	20.64	19.88	31.64	21.27	6.14	6.85	20.53	18.69	26.68	19.63
	0.01	11.01	9.66	18.32	15.45	29.33	18.12	5.59	5.99	20.35	18.36	25.94	19.26
	0.05	8.87	6.60	13.45	10.56	22.31	11.53	5.49	5.78	20.85	17.71	26.34	18.53
	0.1	9.26	6.95	12.95	9.81	22.21	11.12	4.91	5.24	21.83	17.85	26.73	18.34
	0.2	9.00	6.67	13.21	9.50	22.21	10.61	4.42	3.94	22.21	18.59	26.63	18.97
	0.3	8.66	6.54	12.89	9.59	21.55	10.50	4.39	3.88	22.11	18.34	26.50	18.75
	0.4	8.80	6.61	13.38	9.80	22.18	10.82	4.52	4.05	22.11	17.50	26.63	18.00
	0.5	8.76	6.41	13.57	10.29	22.33	11.10	4.44	4.30	22.22	17.68	26.66	18.12

表 10: 联通计费服务测试结果

Strategy		Partition Schema 1						Partition Schema 2					
		F	SD_F	NF	SD_{NF}	T	SD_T	F	SD_F	NF	SD_{NF}	T	SD_T
RT		20.17	16.32	16.50	13.20	252.80	191.54	20.17	16.32	16.50	13.20	252.80	191.54
RPT		17.71	16.78	16.72	25.12	178.91	147.96	15.64	13.4	13.97	22.24	175.00	126.88
DRT	1.0E-5	20.83	21.16	26.81	28.42	2521.96	1866.25	21.88	20.52	26.81	26.52	4166.31	2708.14
	5.0E-5	21.14	20.89	25.50	27.22	2523.77	1808.06	21.39	20.49	26.01	26.23	4188.00	2798.73
	1.0E-4	21.64	20.66	26.82	26.59	2367.56	1574.01	21.60	20.64	24.80	24.45	4077.07	2668.71
	5.0E-4	21.29	21.22	28.37	30.68	2538.89	1882.89	22.09	22.20	25.23	26.04	4073.15	2518.34
	0.001	20.74	19.64	26.60	28.85	2449.44	1615.91	21.95	21.01	25.35	25.45	4224.83	2582.71
	0.005	20.95	19.68	25.96	27.18	2487.14	1778.65	21.69	21.89	27.02	27.24	4215.41	2991.54
	0.01	21.54	20.94	26.42	27.05	2579.27	1809.45	21.39	19.16	26.56	27.54	3991.68	2377.37
	0.05	22.07	21.08	25.37	27.13	2662.38	1931.18	21.46	21.95	24.56	25.70	4005.80	2557.51
	0.1	21.34	21.39	27.14	28.41	2362.79	1552.16	22.62	22.79	25.63	25.92	3966.92	2371.21
	0.2	21.75	20.10	25.39	26.44	2474.91	1780.26	23.33	21.67	26.15	25.70	3992.59	2554.38
	0.3	21.94	21.36	25.93	25.58	2515.45	1860.04	21.01	21.24	25.49	26.54	4068.50	2448.81
	0.4	22.14	21.45	26.39	27.45	2486.92	1732.76	20.04	22.77	27.37	29.88	4045.44	2329.75
	0.5	20.85	19.91	26.33	28.81	2450.95	1703.39	20.42	22.09	25.62	26.15	4097.77	2598.52

表 11: 停车计费服务测试结果

Strategy		Partition Schema 1						Partition Schema 2					
		F	SD_F	NF	SD_{NF}	T	SD_T	F	SD_F	NF	SD_{NF}	T	SD_T
RT		13.30	10.34	28.50	23.95	41.80	27.13	13.30	10.34	28.50	23.95	41.80	27.13
RPT		11.93	11.24	24.36	22.56	35.99	25.02	8.59	8.02	22.47	19.97	27.06	18.73
DRT	1.0E-5	16.93	17.55	16.38	23.38	169.63	136.31	15.08	13.62	12.91	21.87	173.13	133.79
	5.0E-5	17.03	19.15	14.77	22.93	174.06	141.08	16.40	14.88	11.43	19.18	172.53	135.13
	1.0E-4	15.45	15.03	16.09	24.73	167.90	137.66	15.73	14.11	12.73	19.52	168.39	131.70
	5.0E-4	16.85	17.13	15.39	22.63	174.46	142.88	16.23	14.74	13.12	21.00	171.36	127.36
	0.001	16.07	16.99	13.66	21.23	177.75	149.88	15.55	13.66	13.46	21.85	174.54	125.99
	0.005	17.61	18.54	15.26	23.70	179.77	157.27	15.79	15.05	12.91	21.63	180.89	133.50
	0.01	17.08	19.09	17.36	27.05	173.48	139.33	15.93	14.79	14.03	22.23	173.45	132.78
	0.05	17.98	17.27	15.37	23.91	174.42	142.11	16.35	14.43	13.12	22.20	169.50	123.41
	0.1	17.73	18.47	15.80	22.54	176.90	141.45	15.05	13.12	12.39	20.52	168.55	127.52
	0.2	17.83	18.24	14.87	23.25	178.63	140.00	15.39	14.23	13.77	21.71	176.68	140.26
	0.3	16.95	17.13	14.46	21.82	175.69	143.24	15.60	14.83	12.81	20.88	172.45	128.59
	0.4	16.74	17.07	14.67	23.10	168.32	137.52	15.59	13.94	13.36	20.81	171.18	125.94
	0.5	18.54	18.56	14.88	23.07	178.21	138.59	15.53	14.44	13.09	21.77	172.31	128.52

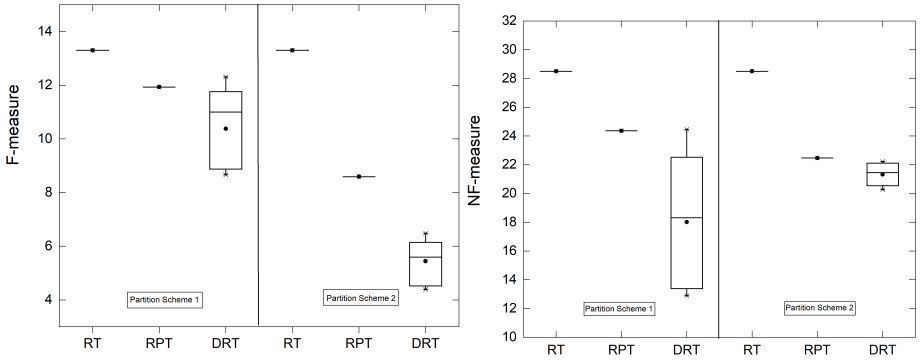
6 实验结果

6.1 RQ1:故障检测效率

每一个实验对象的 F-measure、NF-measure 以及 T-measure 的结果分别记录在表 9 10 11中。每一个实验对象中DRT策略在不同参数下 F-measure、NF-measure 以及 T-measure 的分布分别展示在图 5 6 7 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。盒子中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。盒子下面胡须的最

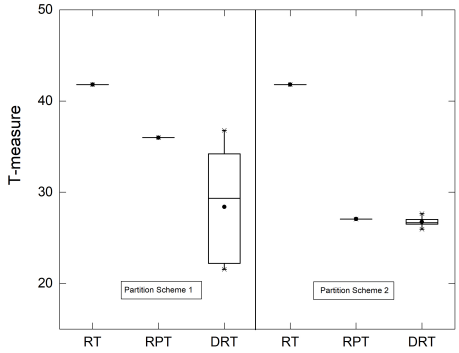
表 12: F-measure 的 Holm 方法统计结果

	RT	RPT	DRT
RT	—	**4**	**71**
RPT	**2**	—	**63**
DRT	**7**	**15**	—



(a) 航空托运服务F-measure

(b) 航空托运服务NF-measure



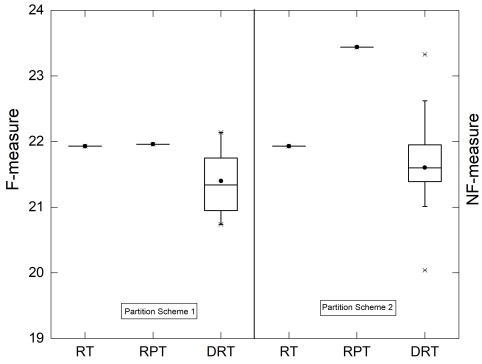
(c) 航空托运服务T-measure

图 5: 航空托运服务结果

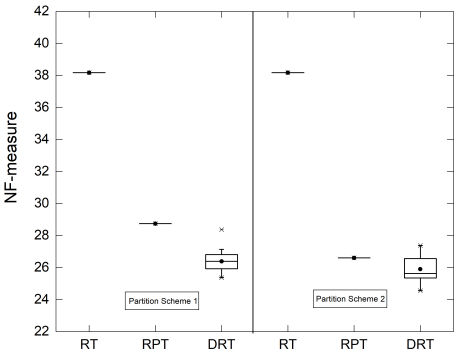
小值为下四分位数 $-1.5 \times IQR$, 上面胡须的最大值为上四分位数 $+1.5 \times IQR$，其中 IQR 表示盒子的长度。不在两个胡须范围内的数值称为异常值。

从表9-11以及图5-7可以看出，表现最好的是DRT策略，RPT策略在多数情况下比RT策略具有更高的故障检测效率。为了验证这一观察结果，本文用 Hlom-Bonferroni 方法($p_{value} = 0.05$) [14] 说明哪一种方法在揭示故障方面表现的更好。统计结果展示在表 12 13 14 中。表中的每一个单元格数值表示该单元格所在列对应的技术比该单元格行对应的技术表现好的频率。如果两种技术的差别很大，对应的统计数值就很醒目。例如表 12右上角的 71 表示在 78 个场景下(3个实验对象，每一个实验对象DRT策略有13个参数并且有两种分区方式) 71 个场景中DRT 策略比RT策略表现更好。由此可以明显分辨出两种测试技术的的故障检测能力。

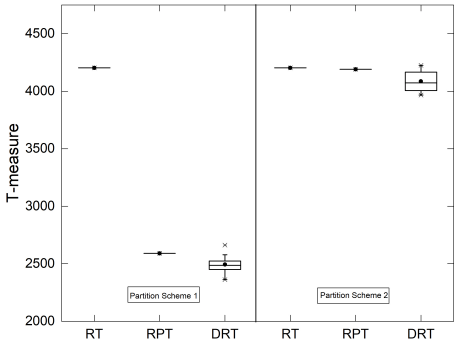
从表 12 13 14 中一方面可以看出不同测试技术的故障检测能力差别很大：DRT策略最好其次是RPT策略。另一方面表 13 14 中的最后一列相对于表 12的最后一列单元格中的数值更醒目。由于揭示故障的测试用例趋向于集簇在连续的区域，DRT策略在揭示第一个故障之后调整测试剖面使得具有更高失效率的



(a) 联通计费服务F-measure



(b) 联通计费服务NF-measure



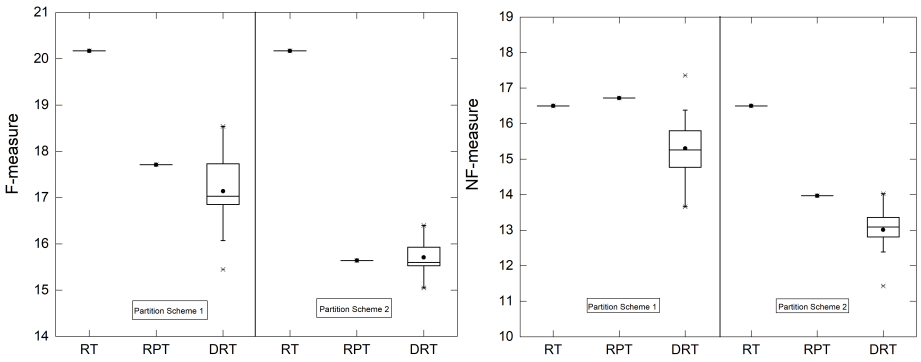
(c) 联通计费服务T-measure

图 6: 联通计费服务结果

分区被选择的概率增大。因此在 NF-measure 和 T-measure 度量指标下DRT策略在故障检测能力方面优势更加明显。

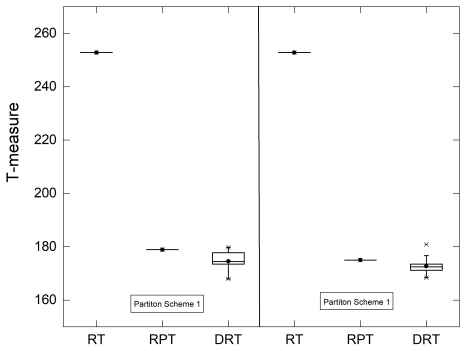
6.2 RQ2:分区数目对DRT策略效率的影响

通过对比每一个图 8、9、10 中两种分区方式下的 F-measure 、NF-measure 、 T-measure 度量指标的大小，我们发现 DRT 策略在对航空行李托运服务进行测试时分区模式 2 更容易揭示第一个故障，分区方式 2 在参数较小时比分区方式 1 更容易揭示故障第二个和所有的故障，但是随着参数的更大，DRT策略在分区方式 1 下更容易揭示故障；DRT 策略在对联通计费服务进行测试时在 F-measure 、 NF-measure 度量指标下两种分区方式几乎没有差别，但是在揭示所有故障方面分区模式 1 更占优势。其原因在于：一，分区模式 1 相当于在分区模式 2 的基础上更细致地划分了分区，使得具有故障的分区失效率变大；二，失效率大的分区能够揭示软件中的多个故障，由于 DRT 测试的内部机制使得揭示故障更加容易；DRT 策略在对停车计费服务进行测试时在 F-measure 、NF-measure 度量指标下分区方式 2 更容易揭示软件中的故障。但是DRT 策略



(a) 停车计费服务F-measure

(b) 停车计费服务NF-measure



(c) 停车计费服务T-measure

图 7: 停车计费服务结果

在 T-measure 度量指标下两种分区方式的故障检测能力很相近。

6.3 RQ3:ε 的值对DRT策略效率的影响

三个实验DRT策略在不同参数值下的情况如图 8 9 10所示。为了方便表示图中的横坐标 $1-13$ 与序列$[0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5]$一一对应。

从图 8中我们可以观察到，在分区模式一中，ε的值从 0.001 到 0.05 DRT 策略的故障检测能力剧烈上升，之后趋于平稳；在分区模式二中，DRT策略在不同的度量标准下故障检测能力趋于平稳。

从图 9中可以看出两种分区方式DRT策略在 F-measure、NF-measure 度量指标下的故障检车能力随着ε 参数值的增大呈现出相似的规律，并且故障检测能力相近。

从图 10中可以看出两种分区方式DRT策略在 F-measure、NF-measure、T-measure 度量指标下的故障检车能力随着ε 参数值的增大趋向于在某一个固定值上上下波动。

表 13: NF-measure 的 Holm 方法统计结果

	RT	RPT	DRT
RT	—	**5**	**77**
RPT	**1**	—	**72**
DRT	**1**	**6**	—

表 14: T-measure 的 Holm 方法统计结果

	RT	RPT	DRT
RT	—	**6**	**76**
RPT	**0**	—	**69**
DRT	**2**	**9**	—

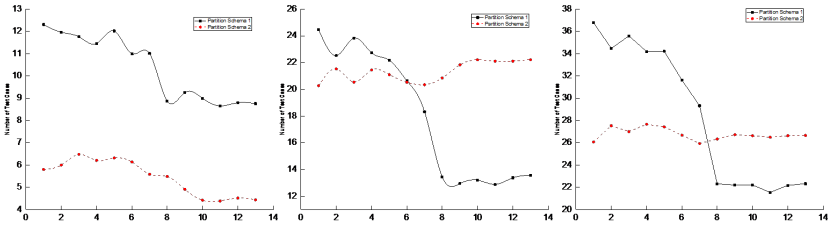
综上所述，我们可以发现在不同的测试对象中 DRT 策略的ε值对其故障检测能力很大的影响，但是很难说某一个值在所有的实验对象下均表现良好。一般情况下我们设置 $\varepsilon = 0.05$ 或者根据Lv在 [17]中提到的方法计算 ε 的理论最佳值，但是该方法需要知道每一个分区的失效率，一些情况下会面临分区的失效率很难得到的情况。

6.4 RQ4:时间开销

三个实验中揭示所有的故障所需要的时间记录在表 15中。

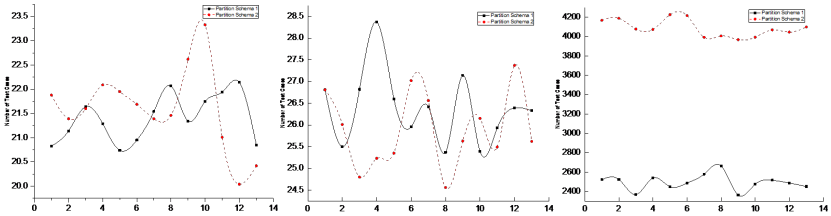
本文本文用 Hlom-Bonferroni 方法比较不同测试策略在揭示所有故障时在时间开销方面的差异，如表 16。

从表 16中可以看出当揭示所有故障时 DRT 策略在时间开销方面比RT测以及RPT策略稍好一点。结合表 15和表 9 10 11进行分析，我们可以看出当测试所需要的测试用例数目较少时RT策略的时间花费时最少的，但是随着测试用例数目的增加 DRT 策略由于揭示所有故障需要的测试用例数目更少，因此时间开



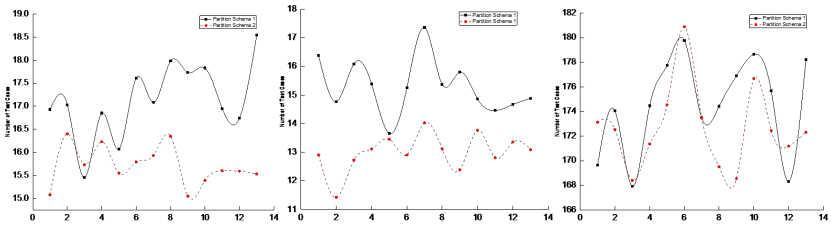
(a) F-measure结果 (b) NF-measure结果 (c) T-measure结果

图 8: 航空托运服务DRT策略不同参数的实验结果



(a) F-measure结果　(b) NF-measure结果　(c) T-measure结果

图 9: 联通计费服务DRT策略不同参数的实验结果



(a) F-measure结果　(b) NF-measure结果　(c) T-measure结果

图 10: 停车计费服务DRT策略不同参数的实验结果

销方面更占优势。

表 15: 揭示所有故障不同测试技术的时间开销(ms)

策略		航空托运		联通计费		停车计费	
		分区方式1	分区方式2	分区方式1	分区方式2	分区方式1	分区方式2
RT		134.46	134.46	1840.95	1840.95	312.59	312.59
RPT		136.37	137.17	1822.61	1842.82	316.86	296.30
DRT	1.0E-5	133.45	136.09	1819.98	1849.69	286.14	276.63
	5.0E-5	140.32	136.10	1828.43	1851.92	282.34	277.63
	1.0E-4	137.64	135.98	1824.62	1850.57	282.24	276.23
	5.0E-4	141.41	135.65	1827.25	1847.15	283.44	276.76
	0.001	137.00	134.78	1823.14	1852.55	282.35	270.11
	0.005	140.48	136.77	1826.27	1851.10	283.35	270.97
	0.01	135.12	140.82	1827.57	1843.91	279.14	273.63
	0.05	140.64	137.08	1829.38	1846.34	281.37	275.68
	0.1	137.25	140.75	1824.03	1843.30	276.93	270.84
	0.2	140.40	136.04	1821.24	1844.62	279.22	283.75
	0.3	136.12	135.78	1826.14	1847.33	279.28	273.53
	0.4	135.62	135.77	1826.52	1850.78	278.75	269.34
	0.5	136.80	135.83	1820.43	1849.51	279.22	276.84

表 16: 测试时间的 Holm 方法统计结果

	RT	RPT	DRT
RT	—	**2**	**40**
RPT	**4**	—	**44**
DRT	**38**	**34**	—

参考文献

[1] Paul E Ammann and John C Knight. Data diversity: An approach to software fault tolerance. *IEEE Transactions on Computers*, 37(4):418–425, 1988.

[2] K. Y. Cai. Optimal software testing and adaptive software testing in the context of software cybernetics ☆. *Information and Software Technology*, 44(14):841–855, 2002.

[3] Kai Yuan Cai, Bo Gu, Hai Hu, and Yong Chao Li. Adaptive software testing with fixed-memory feedback. *Journal of Systems & Software*, 80(8):1328–1348, 2007.

[4] Kai Yuan Cai, Tao Jing, and Cheng Gang Bai. Partition testing with dynamic partitioning. In *Computer Software and Applications Conference, 2005. COMPSAC 2005. International*, pages 113–116 Vol. 1, 2005.

[5] Gerardo Canfora and Massimiliano Penta. Service-oriented architectures testing: A survey. *Software Engineering*, 5413:78–105, 2006.

[6] Gerardo Canfora and Massimiliano Di Penta. *Service-Oriented Architectures Testing: A Survey*. Springer Berlin Heidelberg, 2009.

[7] T. Y. Chen and Y. T. Yu. On the relationship between partition and random testing. *Software Engineering IEEE Transactions on*, 20(12):977–980, 1994.

[8] Tsong Yueh Chen, Pak Lok Poon, and Xiaoyuan Xie. Metric: Metamorphic relation identification based on the category-choice framework ☆. *Journal of Systems & Software*, 116:0000, 2016.

[9] Tsong Yueh Chen and Yuen Tak Yu. *On the Expected Number of Failures Detected by Subdomain Testing and Random Testing*. IEEE Press, 1996.

[10] George B. Finelli. Nasa software failure characterization experiments. *Reliability Engineering & System Safety*, 32(1 ‐ 2):155–169, 1991.

[11] Walter J. Gutjahr. Partition testing vs. random testing: The influence of uncertainty. *IEEE Transactions on Software Engineering*, 25(5):661–674, 1999.

[12] D. Hamlet and R. Taylor. Partition testing does not inspire confidence. In *The Workshop on Software Testing*, pages 206–215, 1990.

[13] Richard Hamlet. Random testing. *Encyclopedia of Software Engineering*, pages 970–978, 1994.

[14] Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

[15] A. Güne? Koru, Khaled El Emam, Dongsong Zhang, Hongfang Liu, and Divya Mathew. Theory of relative defect proneness. *Empirical Software Engineering*, 13(5):473–498, 2008.

[16] Ye Li, Bei Bei Yin, Junpeng Lv, and Kai Yuan Cai. Approach for test profile optimization in dynamic random testing. In *Computer Software and Applications Conference*, pages 466–471, 2015.

[17] Junpeng Lv, Hai Hu, and Kai-Yuan Cai. A sufficient condition for parameter estimation in dynamic random testing.

[18] Junpeng Lv, Hai Hu, and Kai Yuan Cai. A sufficient condition for parameters estimation in dynamic random testing. In *Computer Software and Applications Conference Workshops*, pages 19–24, 2011.

[19] Timothy I Murphy. Line spacing in latex documents. http://www.bcia.com.cn/server/notice/package.shtml. Accessed December 25, 2017.

[20] Timothy I Murphy. Line spacing in latex documents. https://baike.baidu.com/item/%E8%81%94%E9%80%9A3G/5882691?fr=aladdin. Accessed December 25, 2017.

[21] MICHAEL P. PAPAZOGLOU, PAOLO TRAVERSO, SCHAHRAM DUSTDAR, and FRANK LEYMANN. Service-oriented computing: A research roadmap. *International Journal of Cooperative Information Systems*, 17(02):223–255, 2008.

[22] Chris Peltz. Web services orchestration: A review of emerging technologies. 2003.

[23] Chang Ai Sun, Guan Wang, Kai Yuan Cai, and Tsong Yueh Chen. Towards dynamic random testing for web services. In *Computer Software and Applications Conference*, pages 164–169, 2012.

[24] Elaine J Weyuker and Bingchiang Jeng. Analyzing partition testing strategies. *IEEE Transactions on Software Engineering*, 17(7):703–711, 1991.

[25] Zijiang Yang, Beibei Yin, Junpeng Lv, Kaiyuan Cai, Stephen S. Yau, and Jia Yu. Dynamic random testing with parameter adjustment. In *Computer Software and Applications Conference Workshops*, pages 37–42, 2014.