BPEL程序的变异测试优化技术与集成化支持工具研究

贾婧婷

北京科技大学

**密　　　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_**

论文题目：行为模型驱动的服务组合程序

测试用例生成技术研究

S20150750

学　 　号：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

贾婧婷

作　 　者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

软件工程

专 业 名 称：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2017年09月29日

行为模型驱动的服务组合程序

测试用例生成技术研究

Research on Optimization Techniques of Mutation Testing for BPEL Programs and Their Supporting Tool

研究生姓名：贾婧婷

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： Jia Jingting

Supervisor： Sun Changai

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：\_\_\_\_\_\_\_\_\_\_\_\_ 密　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

004.41

TP311

１０００８

ＵＤＣ：\_\_\_\_\_\_\_\_\_\_\_\_ 单位代码：\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**北京科技大学硕士学位论文**

行为模型驱动的服务组合程序测试用例生成技术研究

**论文题目：**

贾婧婷

**作者：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

北京科技大学

孙昌爱 教授

**指 导 教 师： 单位：**

**指导小组成员： 单位：**

**指导小组成员： 单位：**

**论文提交日期：**2016年 12月 21日

**学位授予单位：北 京 科 技 大 学**

致 谢

时光飞逝，两年半的研究生生活很快就要划上句点。在这期间，我的知识水平和实践能力上了一个新的台阶，各方面的素质得到了很大的提高，当然这离不开我的导师孙昌爱老师的耐心指导与教育。

首先，衷心感谢我的导师孙昌爱老师对本次毕业设计的耐心指导与大力支持。本文的研究工作从选题、开题、中期及最后的论文修改等各个方面都离不开孙老师耐心的帮助和教导。其次，研究生期间，孙老师一丝不苟的工作态度，宽厚的生活作风，都给我留下了深刻的印象，对我孜孜不倦的耐心教导，使我取得了进步，也为以后的工作和生活打下了基础。

同时感谢在硕士阶段所有教过我的老师，从他们身上我领略到学术大师的风采，研究人员该有的严谨态度。也要感谢在我的研究阶段帮助过我的王巧玲师姐和王真师妹，感谢他们对我的无私帮助。

最后，非常感谢我的父母，是她们始终站在我背后给予我无私的爱，精神上不断的鼓励，物质上不断的支持才使得我不断前进。

摘 要

在云计算环境中越来越多的应用程序部署为Web服务，为了实现复杂的功能，通常需要将多个Web服务协调组织起来。由于Web服务的实现透明、描述文档信息有限、接口与实现的不一致等特点，使得服务的使用者无法完全了解Web服务的正确使用方式，从而在调用异常时无法分辨究竟是Web服务本身的错误，还是使用Web服务的方法有误。

服务的使用者依据WSDL访问相关Web服务，WSDL是一种基于XML格式的Web服务描述语言，仅仅包含Web服务接口的抽象描述，而缺乏对服务提供操作的语义信息。**为了确保服务调用时的正确性和可靠性**，本文从Web服务行为相关的**数据**和**控制约束**出发，提出了行为模型驱动的Web服务组合测试用例生成技术，并开发了相应的支持工具。向WSDL文档中添加定义满足各种约束的覆盖准则的测试用例生成算法，调用web服务提供的操作，对其进行测试及监控，有效地监控运行时刻可能出现的异常（可能会去掉）。本文取得的主要成果如下：

1. **扩展WSDL支持服务行为约束的表达**，通过研究Web服务提供操作自身及相互之间可能存在的约束，向Web服务描述语言(WSDL)中添加数据与控制流方面的约束描述；
2. **提出了一种基于扩展WSDL的服务行为模型生成技术：**解析扩展后的WSDL，采用事件序列图对Web服务进行行为模型的建立；
3. **提出了一种行为模型驱动的Web服务测试用例生成技术：**在给定的行为模型的基础上，依据服务行为模型的特征，定义各种**覆盖准则**，设计相应的**测试序列生成算法**，针对每条测试序列生成**满足约束的测试数据**，将测试数据与测试序列相结合，生成可执行的**测试用例**；
4. **开发了行为模型驱动的Web服务组合测试用例生成工具BD\_WSTCT：**BD\_WSTCT支持扩展后服务描述文档（EX-WSDL）解析、服务行为模型生成及可视化、测试路径集生成、测试数据生成、测试用例集生成、测试执行及判定和测试结果统计；
5. **经验研究：**采用三个Web服务程序实例验证并评估提出的技术的有效性与支持工具的实用性。

本文提出的行为模型驱动的Web服务组合测试用例生成技术，可以有效的XXX。开发的支持工具提高了Web服务测试用例生成的自动化程度，进一步提了xxx的效率。

关键词： 行为模型，WSDL，Web服务约束，测试用例

Research on Optimization Techniques of Mutation Testing for BPEL Programs and Their Supporting Tool

Abstract

Distributed applications are increasingly deployed as Web services in the cloud computing environment. Multiple services are normally composed to fulfill complex functionalities. Business Process Execution Language (BPEL) is an XML-based service composition language that is widely used to define a complex business process by orchestrating multiple services. Due to unique features of Web services, such as dynamics, loose coupling, and open deployment and execution environment, it is an important issue how to assure the quality of Web services and their compositions. Mutation testing is a kind of fault-based software testing technique that has been widely used for evaluating the adequacy of test suites and the fault-detection effectiveness of software testing methods. However, mutation testing failed to be widely practiced because it normally generates a large amount of mutants and thus incurs a high cost.

In order to improve the practicability of the mutation testing technique, this thesis studies how to decrease the cost of mutation testing for BPEL programs. We propose a series of optimization techniques in terms of subsuming relation of mutation operators, the prioritization of operators, and second-order mutation, and develop a tool named μBPEL to implement these optimization techniques. The main contributions made in this thesis are as follows:

1. **Three optimization techniques of mutation testing for BPEL programs**, including a subsuming relation based optimization technique that is based on the analysis of mutation operators and the subsuming relation among operators, an operator priority based optimization technique that is based on ranking of mutation operators using a quality merit, and a second-order mutant optimization technique for BPEL programs.
2. **An integrated mutation testing system for BPEL programs called μBPEL,** which supports the whole lifecycle of the mutation testing for BPEL programs, including mutant generation, optimization techniques, test case generation, test execution, and analysis of testing results.
3. **An empirical study** in which six representative BPEL programs are used to validate and evaluate the effectiveness of the proposed optimization techniques.

In summary, the proposed optimization techniques of mutation testing for BPEL programs are able to reduce the number of mutants without significantly jeopardizing its fault detection effectiveness. The supporting tool further improves the automation [of](E:/tool/Youdao/Dict/6.3.69.8341/resultui/frame/javascript:void(0);) mutation testing and thus increases its efficiency.

Key Words： Service Compositions, BPEL, Mutation Testing,

Testing Tool, Subsuming Relation

目 录

[致 谢 I](#_Toc470079449)

[摘 要 III](#_Toc470079450)

[Abstract V](#_Toc470079451)

[1 引言 1](#_Toc470079452)

[1.1 研究背景与意义 1](#_Toc470079453)

[1.2 研究内容与成果 2](#_Toc470079454)

[1.3 论文组织结构 3](#_Toc470079455)

[2 背景介绍 4](#_Toc470079456)

[2.1 相关概念与技术 4](#_Toc470079457)

[2.1.1 BPEL 7](#_Toc470079458)

[2.1.2 变异测试 10](#_Toc470079459)

[2.1.3 BPEL程序的变异算子 **错误!未定义书签。**](#_Toc470079460)

[2.2 国内外研究现状 11](#_Toc470079461)

[2.2.1 变异测试优化技术研究现状 11](#_Toc470079462)

[2.2.2 BPEL测试研究现状 **错误!未定义书签。**](#_Toc470079463)

[2.2.3 课题组相关工作 14](#_Toc470079464)

[3 面向BPEL程序的变异测试优化技术 16](#_Toc470079465)

[3.1 基于包含关系的变异测试优化技术MuSR 18](#_Toc470079466)

[3.1.1 方法原理 18](#_Toc470079467)

[3.1.2 一般过程 22](#_Toc470079468)

[3.2 基于二阶变异体的变异测试优化技术MuSOM 22](#_Toc470079469)

[3.2.1 方法原理 22](#_Toc470079470)

[3.2.2 一般过程 22](#_Toc470079471)

[3.2.3 方法示例 22](#_Toc470079472)

[3.3 基于变异算子优先级的变异测试优化技术MuPri 22](#_Toc470079473)

[3.3.1 方法原理 22](#_Toc470079474)

[3.3.2 一般过程 22](#_Toc470079475)

[3.3.3 方法示例 22](#_Toc470079476)

[3.4 小结 22](#_Toc470079477)

[4 面向BPEL程序的变异测试集成化工具设计与实现 22](#_Toc470079478)

[4.1 需求分析 22](#_Toc470079479)

[4.2 μBPEL设计与实现 22](#_Toc470079480)

[4.2.1 系统架构 22](#_Toc470079481)

[4.2.2 工具实现 22](#_Toc470079482)

[4.3 系统演示 22](#_Toc470079483)

[4.4 小结 22](#_Toc470079484)

[5 经验研究 22](#_Toc470079485)

[5.1 实验对象与度量指标 22](#_Toc470079486)

[5.2 扩充的实例研究 22](#_Toc470079487)

[5.2.1 实验步骤 22](#_Toc470079488)

[5.2.2 实验结果 22](#_Toc470079489)

[5.3 变异测试优化技术实例评估 22](#_Toc470079490)

[5.3.1 MuSR技术评估 22](#_Toc470079491)

[5.3.2 MuSOM技术评估 22](#_Toc470079492)

[5.3.3 MuPri技术评估 22](#_Toc470079493)

[5.4 小结 22](#_Toc470079494)

[6 工作总结与展望 22](#_Toc470079495)

[参考文献 22](#_Toc470079496)

[作者简历及在学研究成果 22](#_Toc470079497)

[独创性说明 22](#_Toc470079498)

[关于论文使用授权的说明 22](#_Toc470079499)

[学位论文数据集 22](#_Toc470079500)

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. 研究背景与意义

面向服务的架构SOA（Service-Oriented Architecture）逐渐成为开发应用程序的主要范式[[[1]](#endnote-1)]。Web服务作为SOA概念的一种典型的实现方式，采用XML标准，使用简单对象访问协议（SOAP）进行通信、网络服务描述语言（WSDL）进行服务描述、UDDI进行服务的注册和搜索[2]，通常由一个服务描述文件WSDL[[[2]](#endnote-2)]定义其提供的操作接口，并由某种开发语言实现上述接口的业务逻辑，Web服务对外提供统一的调用接口，而屏蔽实现细节[[[3]](#endnote-3)]。对于复杂多变的企业级业务需求而言，单一的Web服务往往无法满足实际需求，需要将多个服务以松耦合的方式协调组织起来以支持复杂应用[[[4]](#endnote-4)]。对于Web服务组合来说，其正确性不仅仅取决于组合流程，也取决于组成它的各个服务[[[5]](#endnote-5)]。由于Web服务及其协同的动态性，松耦合的服务开发模式，使得服务使用者或系统集成商在调用Web服务时容易出现异常：

各个Web服务相互协作的动态性，互联网环境的开放性，Web服务组合的松耦合性所导致的组合和运行Web服务的不确定性[3]，对Web服务及其服务组装的质量保证带来挑战。

1. **实现透明：**对服务使用者和系统集成商，服务只是提供了接口，实现部分是透明的[[[6]](#endnote-6)]；
2. **服务描述文档信息有限：**WSDL仅包含Web服务接口的抽象描述，服务的使用者并不能了解操作中潜藏的数据与控制流方面的约束；
3. **接口与实现的不一致：**需求的快速变化导致参与组装的服务面临经常性的修改，服务的接口与实现之间潜藏不一致性。

上述特点使得服务的使用者或者系统集成商无法完全了解Web服务的正确使用方式。从而在调用过程中无法分辨究竟是Web服务本身的错误，还是使用Web服务的方法有误。

其中，该问题主要是由Web服务的WSDL文件提供的信息不足引起的。WSDL是一种基于XML格式的Web服务描述语言，仅仅包含Web服务接口的抽象描述，而缺乏对服务提供操作的语义信息[[[7]](#endnote-7)]。

由于服务的使用者只能依据WSDL访问相关Web服务，**为了确保服务调用时用户调用的正确性**，我们考虑向WSDL中添加服务行为相关约束，并自动生成 辅助服务调用者更直观的了解服务运行过程中的约束，生成测试

软件测试是保证服务质量和可靠性的必要手段。然而，由于SOA软件的上述特点，使得已有的基于程序的测试方法很难直接用来测试Web服务。现有研究提出将模型驱动的测试技术应用于Web服务。模型驱动的测试关注待开发系统的重要属性及约束，具有高故障检测率，且自动化程度高，能够更好的适应需求的演变，能够较好的适应Web服务的测试需求，因此越来越多的研究人员从模型角度出发对Web服务及其组合进行测试。

然而现有模型驱动web服务测试的研究**主要针对于单个服务层面，或是服务组合流程的测试**，集中在模型驱动的测试用例生成框架的研究，**缺乏支持所提框架的测试用例生成工具**；同时基于事件序列的测试模型**缺乏对调用服务内部约束的考虑**。

因此，本文从服务调用视角对Web服务的行为进行测试，在事件模型中考虑Web服务行为相关的数据和控制约束，提出了行为模型驱动的Web服务组合测试用例生成技术，并开发了相应的支持工具辅助所提技术的测试用例自动化生成。

* 1. 研究内容与成果

本文从Web服务行为相关的数据和控制约束出发，提出了行为模型驱动的Web服务组合测试用例生成技术，并开发了相应的支持工具。主要研究内容与成果如下：

1. **扩展WSDL支持服务行为约束的表达**，通过阅读文献研究总结Web服务提供操作自身及相互之间可能存在的约束，利用WSDL扩展机制向Web服务描述语言(WSDL)中添加数据与控制流方面的约束描述；
2. **提出了一种基于扩展WSDL的服务行为模型生成技术：**解析扩展后的WSDL，采用事件序列图对Web服务进行行为描述模型的建立，使用Graphviz进行行为模型可视化；
3. **提出了一种行为模型驱动的Web服务组合测试用例生成技术：**在给定的行为模型的基础上，依据服务行为描述模型的特征，定义各种**覆盖准则**，设计相应的**测试序列生成算法**；针对每条测试序列**自动**生成**满足约束的测试数据**，设计相应的测试数据与测试序列合成算法，生成可执行的**测试用例**；
4. **开发了行为模型驱动的Web服务组合测试用例生成工具BD\_WSTCT:** BD\_WSTCT支持测试的全过程，支持扩展后服务描述文档解析、服务行为模型生成及可视化、测试序列自动生成、测试数据自动生成、测试用例集生成、测试执行、测试结果判定及统计；
5. **经验研究：**采用三个Web服务程序实例验证并评估提出的技术的有效性与支持工具的实用性。

本文研究工作得到了北京市自然科学基金面上项目“模型驱动的SOA软件测试与监控技术研究”（项目编号：XXX）的资助。

* 1. 论文组织结构

本文的组织结构安排如下：

1. 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。
2. 背景介绍，主要包括相关概念及技术、国内外研究现状。
3. 深入讨论行为模型驱动的服务组合程序测试用例生成技术，主要包括对技术的基本思想、原理描述及具体实现。
4. 讨论行为模型驱动的服务组合程序测试用例生成工具的设计与实现，包括需求分析、系统设计与实现以及系统演示。
5. 对Web服务实例进行经验研究，验证测试用例生成技术的可行性与有效性，主要包括研究问题阐述、实验对象的说明、实验设计与过程、度量指标以及实验结果分析。
6. 研究工作的总结以及未来的展望。
7. 背景介绍

本章介绍本文涉及到的相关技术、工具实现以及实验涉及到相关工具、国内外WSDL扩展研究及Web服务模型测试的相关进展。

* 1. 相关概念与技术

介绍论文中涉及的相关技术及工具，包括Web服务、模型测试等相关概念以及模型测试、模型可视化、约束求解等工具。

* + 1. Web服务

Web服务作为SOA的一种实现方式，是一个平台独立、松耦合、自包含、可编程的应用程序，Web服务的描述、访问、检索与发布基于标准XML技术以及一系列的Web技术标准[[[8]](#endnote-8)]。这些技术及标准有效地屏蔽了运行环境的异构性，使得Web服务可以直接部署和运行于Internet之上。Web服务的开发与使用主要包括如下协议标准：

**WSDL（Web服务描述语言）**[2]**：**WSDL是一种基于XML的定义用以描述Web服务的语言，WSDL文件中包含了与服务交互的接口描述，如消息格式，传输协议以及位置等，但不包括Web服务的实现细节。

**UDDI（通用描述、发现和集成）[http://www.uddi.org/pubs/uddi\_v3.htm]:** UDDI是一个由OASIS开发的跨行业的注册标准草案，基于一系列常见的行业标准（HTTP、XML、XML Schema以及SOAP）。定义了一个发布和检索可用Web服务信息的通用机制，可实现Web服务的描述和发现。

**SOAP（简单对象访问协议）**[8]**：**SOAP是一种简单的消息传递协议，提供了标准的调用Web服务的方式，用户发出一个SOAP消息作为服务请求，Web服务经过处理返回一个SOAP消息作为服务的响应。

XML为Web服务各角色之间的交互提供了标准的、平台无关的数据格式。WSDL、UDDI和 SOAP均是基于XML定义的。

* + 1. WSDL

Web服务描述语言[2,[[9]](#endnote-9)]用于定义Web服务的接口，给出请求消息的格式，并告诉服务的使用者采用何种通信协议以及在何处访问Web服务。以XML的形式描述Web服务、参数和返回值。Web服务使用者通过WSDL文档中定义的接口规约、消息格式规约、通信协议规约以及访问地址完成对服务的调用。

WSDL协议将其对服务的描述分为抽象定义和具体实现两部分，抽象定义描述该服务的操作和消息，具体实现则定义了绑定等和具体服务地址相关的信息。抽象定义的主要组成元素为<types>，<message>和<portType>，具体实现则由<binding>和<service>两个元素组成。WSDL文档是由一个或多个WSDL XML信息集表示的，其XML语法及部分标签含义如图2-1所示：



Binding

Message

Service

portType

Types

图2-1 WSDL内容模型

|  |
| --- |
| <wsdl:definitions name="nmtoken"? targetNamespace="uri"?>  <import namespace="uri" location="uri"/>\*  <wsdl:documentation .... /> ? |
| <wsdl:types> ?  <xsd:schema .... />\*  </wsdl:types> |
| <wsdl:message name="nmtoken"> \*  <part name="nmtoken" element="qname"? type="qname"?/> \*  </wsdl:message> |
| <wsdl:portType name="nmtoken">\*  <wsdl:operation name="nmtoken">\*  <wsdl:documentation .... /> ?  <wsdl:input name="nmtoken"? message="qname">?  </wsdl:input>  <wsdl:output name="nmtoken"? message="qname">?  </wsdl:output>  <wsdl:fault name="nmtoken" message="qname"> \*  </wsdl:fault>  </wsdl:operation>  </wsdl:portType> |
| <wsdl:binding name="nmtoken" type="qname">\*  <wsdl:operation name="nmtoken">\*  <wsdl:input> ?  <-- extensibility element -->  </wsdl:input>  <wsdl:output> ?  <-- extensibility element --> \*  </wsdl:output>  <wsdl:fault name="nmtoken"> \*  <-- extensibility element --> \*  </wsdl:fault>  </wsdl:operation>  </wsdl:binding> |
| <wsdl:service name="nmtoken"> \*  <wsdl:documentation .... />?  <wsdl:port name="nmtoken" binding="qname"> \*  <-- extensibility element -->  </wsdl:port>  </wsdl:service> |
| </wsdl:definitions> |

* 类型(Types)：通过某种类型系统（XMLScheme）定义了Web服务消息交换所需的各种数据类型，这些类型将在消息部件中使用。
* 消息(Message)：描述Web服务发送或接收的特定消息的抽象格式，消息由逻辑部分（Part）组成，每个部分与某些类型系统（Types）中的定义相关联。
* 操作(Operation)：描述给定端口类型支持的操作，即具体的消息访问接口，包括输入、输出或故障消息。
* 端口类型(PortType)：某个访问接口类型所支持的操作的抽象集合，通过将相关消息分组到操作（Operation）中用以描述服务发送和/或接收的一组消息，可以包含若干个“Operation”。
* 绑定(Binding)：用来绑定相应的传输协议，它定义的是一种通讯的方式，每一个PortType对应一个Binding，然后在Binding中进一步细化设置每一个操作，配置每一个input, output, fault的传出方式，编码方式等。
* 端口(Port)：定义单个服务访问点的访问入口部署细节。
* 服务(Service)：描述服务提供的端口类型集合及其提供的端口，一个特定的WSDL服务可以关联多个不同的URI或者不同的端口类型。
  + 1. 模型驱动测试技术

模型驱动技术(MDA)和以测试为中心的软件开发技术与方法的兴起和应用以及形式化验证技术的逐步成熟，使基于模型的软件测试方法与技术(MBT)在近几年得到了较广泛的关注[[[10]](#endnote-10),[[11]](#endnote-11)]。基于模型的测试是一种有效的软件测试方法，它关注待测系统(System Under Test)的属性或约束，通过采用形式化或半形式化语言对复杂软件期望行为和环境的可能行为进行抽象与建模，依据模型进行测试的设计[[[12]](#endnote-12)]。

Dalal提出MBT方法的自动化依赖于三个主要元素：软件行为描述模型、测试覆盖算法及针对测试的基础配套设施[[[13]](#endnote-13)]。一个通用MBT过程如图2-2[[[14]](#endnote-14)]所示：

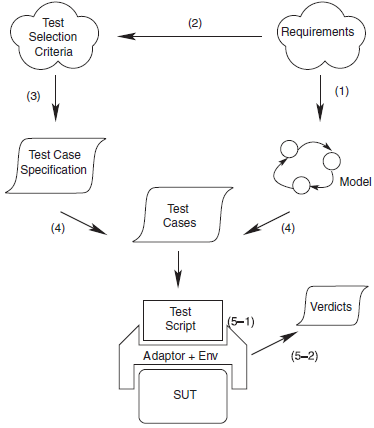


图2-2 一般MBT过程

1. 从待测软件的现有规范文件或需求文件中建立模型：该模型一般称为测试模型，因为模型测抽象测试和焦点直接与测试目标关联，某些情况下，测试模型可以也是待测程序的设计模型；
2. 选择测试准则指导测试用例集生成自动；
3. 将选定的测试准则转换成测试用例规格：规格规范了测试选择标准的概念，并使其针对测试模型具有可操作性，依据测试用例规格可从测试模型中派生出测试用例集；
4. 依据测试用例规格可从测试模型中派生出测试用例集；
5. 测试用例运行：测试的执行可能是手工执行也可能是自动化的测试执行环境（依赖提供自动执行测试和记录测试结果的工具）。
   * 1. 模型测试及模型可视化工具

* 模型测试工具GraphWalker

针对测试序列生成问题，我们集成现有基于模型的测试工具GraphWalker，GraphWalker是一个开源的基于模型的测试用例生成工具，用来解决从有向图模型中生成测试路径[[[15]](#endnote-15)]，主要应用于FSM（有限状态机）、EFSM（扩展有限状态机）模型。该工具可作为一个java库集成到项目中。

GraphWalker提供了一种XML描述的图模型语言(Graph Model Language)以及多种图模型遍历策略。GraphWalker工作流程如图2-3所示，下面将对其核心步骤进行介绍。



图2-3 GraphWalker工作流程图

1. 模型设计：

GraphWalker模型为有向图模型，由node，edge元素组成，使用xml格式表示，其模型语法及部分标签含义如图2-4所示：



图2-4 Graph Model Language语法

图(graph)标签描述GraphWalker模型的抽象格式，包括节点(node)及边(edge)；节点标签描述GraphWalker模型中包含的点，包括节点名称、编号及其他属性；边描述GraphWalker模型节点的转移，包括边名称、编号、及其他属性。

1. 遍历算法选择：

GraphWalker提供系列遍历规则，支持从模型中根据想覆盖的元素导出目标路径。主要解决在生成一个模型的路径时使用什么策略，以及何时停止生成该路径。

1. 测试路径生成：

GraphWalker支持在线和离线两种测试路径生成策略，本文使用离线策略，通用命令行执行相关遍历算法，生成所需测试路径，便后续测试用例生成。

* 模型可视化工具Graphviz

本文使用开源图形可视化软件Graphviz显示生成的行为模型。Graphviz是一个图形绘制工具集，用来绘制结构化的图形网络,支持多种格式输出[[[16]](#endnote-16)]。Graphviz的核心包括各种常见图形布局的实现（dot、twopi、circo、fdp、neato）。这些布局可以通过C库接口、命令行工具、图形用户界面和Web浏览器来使用[[[17]](#endnote-17)]。

本文使用dot布局实现行为模型可视化。dot为Graphviz默认布局方式，主要用于有向图的可视化[[[18]](#endnote-18)]。将生成的行为模型图转换为dot脚本，通过布局引擎来解析dot脚本[[[19]](#endnote-19)]，分析出其中的点，边以及子图，然后根据属性进行绘制。

* + 1. Z3约束求解器

针对测试数据生成问题，人工分析方法耗费时间且随机性较大，因此本文考虑基于自动约束求解(Constraint Solver)工具来解决测试数据的生成的主要问题。对比目前已经出现约束自动求解工具[[[20]](#endnote-20)]，鉴于Z3[[[21]](#endnote-21)]在操作系统、语言支持、求解类型支持（Z3-str支持对String类型约束条件的处理能力，将String类型作为基本类型进行处理，使得求解器的应用范围更加广泛）、开放源代码、求解性能高等优点，我们使用Z3来自动生成所需测试数据。

Z3是一款微软研究院开发的SMT求解器，用于对一阶逻辑表达式组合求出可行解[[[22]](#endnote-22)]，其目标是解决软件验证与分析中出现的问题，例如测试用例生成、程序缺陷检测[[[23]](#endnote-23),[[24]](#endnote-24),[[25]](#endnote-25)]等。因此，它集成了多种理论支持，提供所有基本的数学运算，逻辑运算，关系运算等，可以解决线性、非线性多项式约束。一个基本的Z3结构如图2-5所示[21]。

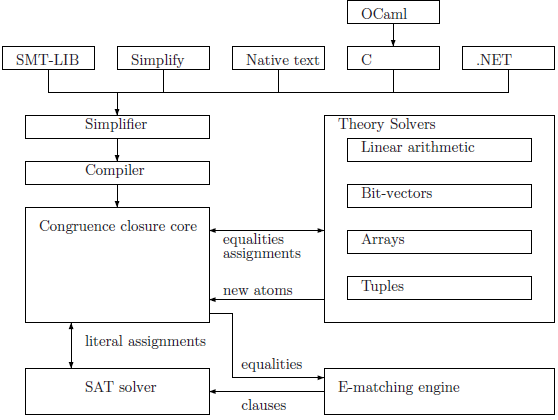


图2-5 Z3结构图

它主要由以下核心模块组成：

* Simplifier：使用简化输入规则对刚输入的程序公式进行一个不完整的，但有效的简化处理。
* Compiler：将简化的抽象语法树表示的公式转换成不同的数据结构，包含不同的子句和同余封闭节点。
* Congruence Closure Core（同余闭包引擎）：同余闭包引擎是Z3核心模块，接收SAT Solver为每个目标产生的真实数据。将Theory Solver和SAT Solver中的节点进行合并处理，判断生成的解是否为可行解。
* SAT Solver（SAT求解器）：用于处理输入公式的逻辑结构，集成了标准搜索修剪方法，这些方法用来搜索可行的解。
* Theory Solvers (理论求解器)：基于算术运算的线性算术求解器，用来解决如线性运算、二进制向量运算、数组运算等可满足性问题的求解器。

Z3作为开源约束求解器，可以作为组件集成到其他需要求解逻辑公式的工具中。Z3针对不同语言提供了不同的API接口，将符合Z3语法规则的脚本编写完成后，调用接口运行Z3求解出满足约束条件表达式的可行解即可。

* 1. 国内外研究现状

本节介绍WSDL扩展技术与基于模型的Web服务测试技术的国内外研究现状与进展。

* + 1. WSDL扩展技术研究现状

针对Web服务的WSDL文件描述信息不足的问题，国内外研究人员尝试从多个角度扩展WSDL。

Tsai[[[26]](#endnote-26)]针对Web服务的WSDL文件问题提供的信息不足的问题从输入输出依赖性(操作输入输出参数关系)，调用序列描述(服务调用关系)，层次功能描述和序列规范(服务执行顺序要求)四个方面扩展WSDL以增强其描述能力。输入输出依赖可识别服务接口间的联系，在回归测试中减少测试用例；服务间调用顺序可捕捉服务间数据流和控制流间的依赖关系；功能层次描述利用层次化结构来描述服务接口的功能。Sheng等人[[[27]](#endnote-27)]引入新的Web服务模型，将Web服务行为分为操作行为和控制行为，操作行为描述了Web服务的业务逻辑，控制行为作为操作行为的控制者引导其Web服务业务逻辑的执行进度，以便更好的测试，分析和调试Web服务。Sneed等人[[[28]](#endnote-28)]提出通过扩展WSDL，增加的描述能力，然后结合树结点、扩展的刻面约束和前置条件来生成相应的测试数据。

针对WSDL文档未提供操作序列生成所需Web服务行为信息的问题，Bertolino等人[[[29]](#endnote-29)]提出服务提供者上载WSDL的同时上载操作序列的协议状态机以描述Web服务行为信息，通过向WSDL中引入UML2中的协议状态机描述，增加对服务前置后置条件以及数据、进程约束的描述，增强了Web服务的可验证性。Heckel等人[[[30]](#endnote-30)]提出服务提供商要上载行为图转换规则来描述Web服务行为信息。但是上述方法缺乏相应的规范，并且增加了服务商的工作量。

针对WSDL描述Web服务非功能属性局限问题，Parimala等人[[[31]](#endnote-31)]从非功能属性角度扩展WSDL的Service标签，向Service中添加criteria name及definition标签，代表目标服务包含的非功能属性及详细说明，更加详细的描述了这个服务提供的服务特性以帮助服务使用者找到更合适其需求的服务。Jiang等人[[[32]](#endnote-32)]通过扩展WSDL文件，向其中增加服务价格、响应时间、服务可用性、可靠性及服务评价等质量因素的方式弥补基于功能属性的Web服务选择的不足。

为了提高服务组合质量，Wang等人[[[33]](#endnote-33)][Constraint-Aware Approach to Web Service Composition]使用WSDL语义注释(SAWSDL)将服务描述分为服务内涵和服务扩展，其中服务内涵描述服务操作及其输入输出参数约束，服务扩展描述服务正确执行的上下文条件（如服务有效时间、有效区域等）及服务的质量参数集。

Heckel R 等人提出来一种基于WSDL描述文档信息的自动的Web服务测试方法，通过解析WSDL描述文档并将其最终转换为DOM树结构，从测试的操作和数据两个方面进行测试用例的生成，最后根据XML的测试规格说明来表示测试用例，但操作间的依赖关系是由输入、输出的参数间的关系定义的，仅仅考虑了Web服务的数据流方面的信息

[Constraints based Web Service Semantic Augmentation]提出了一种基于约束的Web服务语义增强方法，（服务约束 对象约束）关注Web服务中心普遍存在的两类参数约束——服务约束、对象约束——扩充Web服务的语义约束

姜瑛等人[7]结合合约式设计，扩充WSDL语法，提出了一种基于前置条件和后置条件的Web服务测试技术，有助于服务使用者正确的使用Web服务，并在使用出错时分辨错误原因，该技术向WSDL中<operation>标签中添加<precondition>及<postcondition>标签分别标识操作前置及后置条件。

袁雪莉等人[[[34]](#endnote-34)]通过对WSDL operation元素中扩展preop子元素标识执行operation必须执行的前置操作，增强了服务操作序列描述。同时提出了基于WSDL参数权重设置的数据自动生成方法，减少测试用例生成个数，生成但是这种方法需要人工进行权重设置，同时也可能将有用的测试数据舍去。

白晓颖等人对Web服务的WSDL进行语义上的简单分析，采用Dom树

Li等人引入了schema特殊树模型，通过分析操作之间交换的数据消息进行操作序列的生成，而这种操作依赖的方法只能处理简单的数据类型，缺乏具体的数据生成方法和扩展应用。

* + 1. 模型驱动的Web服务测试研究现状

目前，基于模型的测试技术逐渐得到关注，并出现了大量针对Web服务及其组合程序的测试模型建模方法。基于模型的软件测试思想来自于硬件测试，它被广泛的应用到电信交换系统的测试中，当被应用到软件测试时，其过程是首先构建待测试软件的模型及其派生模型（一般称作测试模型），然后从模型中生成需要的测试用例，在待测程序上执行该测试用例得到测试结果[[[35]](#endnote-35)]。

国内外现有研究主要关注Web服务执行时的行为及其内部状态，首先根据Web服务或其组合程序的行为建立有限状态机[[[36]](#endnote-36),[[37]](#endnote-37),[[38]](#endnote-38),[[39]](#endnote-39),[[40]](#endnote-40),[[41]](#endnote-41)][12-17 参考文献不够的话可以看组合Web服务测试方法研究里面的《10 11》]、事件序列图[[[42]](#endnote-42),5,[[43]](#endnote-43),[[44]](#endnote-44)][3,9,19,20]等模型，或者结合UML建模技术[35,[[45]](#endnote-45),[[46]](#endnote-46),[[47]](#endnote-47),[[48]](#endnote-48),[[49]](#endnote-49)][11,21-25]进行模型的建立，然后采取一定的覆盖准则，生成测试路径。

文献[38]提出了一种基于有限状态机FSM[[[50]](#endnote-50)][26]的Web服务建模测试方法及支持工具，该方法专注于服务测试场景中的状态及转移。有限状态机由有限个状态及状态之间的迁移组成，转换条件对应Web服务所执行的操作，即服务的请求及响应；状态对应待测程序目前所处状态。但该方法并未考虑各个状态之间的转移条件（输入、输出数据及程序内部变量值）。

Keum[39]和Kalaji[40]等人提出了一种基于扩展的有限状态机（EFSM）进行Web服务的建模与测试方法，在传统有限状态机模型中添加状态转换的约束（操作输入输出参数要求，变迁的前置条件等）。通过扩展的WSDL文档（向其中添加服务行为约束）进行模型的建立，利用建立的模型生成测试用例。由于EFSM在FSM基础上添加了变量约束，因此它可以对待测服务的控制流和数据流进行建模，但该方法并未考虑服务组合程序的测试。

文献[41]针对如何实现服务组合程序的足够的测试覆盖率的问题提出了面向服务组合的分层有限状态机模型。

有限状态机可以比较精确的刻画软件系统的行为，因此被广泛应用于很多领域模型系统的建立，包括时序电路、程序验证、网络交互协议等。但是有限状态机和状态转换模型不能很好的体现Web服务的动态交互，描述具有典型并发特征的工作流系统的能力较弱，也不利于系统的负面测试。

文献[43]提出了一种基于事件序列图的Web服务测试方法（ESG4WS），测试Web服务的功能行为，事件序列图模型是无状态的，他们不关注软件组件的内部状态，而是专注于事件[[[51]](#endnote-51)][27]。事件序列图由节点及边组成，节点代表服务的请求或响应事件，边代表这些事件的序列，该方法能够有效地进行正面及负面测试。

文献[42]在其基础上添加了基于覆盖的测试方法，实现了控制流和数据流测试覆盖标准，但该研究仅针对单个Web服务且要求被测服务开放源码。

文献[5,44]在文献[43]的基础上，针对服务组合程序进行事件序列模型的建立及测试用例的自动生成，开发了相应的支持工具。

虽然上述模型描述了系统交互过程，但它**忽略了测试中的系统的内部状态**。

也有研究方向考虑结合统一建模语言进行模型的建立，统一建模语言(unified modeling language，简称UML)是一种通用的可视化面向对象建模语言，它提供了多种图元，可以从不同视角和层次描述复杂软件系统的静态结构和动态特性，其定义良好、易于表达，已广泛应用于各种领域[[[52]](#endnote-52)][28]。（基于场景的Web服务组合并行测试生成的研究里面提到UML建模的相关文献较多）

文献[45]提出了一种基于扩展有限状态机及UML顺序图相结合的服务组合程序测试用例生成框架（EFSM-SeTM）并定义了5种覆盖准则（全路径覆盖、状态覆盖、转移覆盖、消息覆盖及谓词覆盖），EFSM-SeTM使用扩展的有限状态机对单个Web服务进行建模、UML顺序图描述服务组合的信息传递顺序。但该工作对测试用例生成的具体实现过程没有做详细介绍。

文献[35,46]提出了基于UML2.0活动图和通信图构建Web服务测试模型进行Web服务测试的方法，该方法扩展了UML2.0活动图，使其能够描述BPEL的语法元素和行为特性；同时，给出UML2.0活动图形式化定义及其测试覆盖准则，对测试用例生成的深度优先搜索算法加以约束，合理地减少了测试用例的数量，提高了测试的效率和精确性。但该方法基于BPEL 的语法元素和行为特性，在应用到其他Web服务组合方式时有其局限性。

现有研究主要主要针对于单个服务层面，或是服务组合流程的测试，集中在模型驱动的测试用例生成框架的研究，缺乏支持所提框架的测试用例生成工具；同时基于事件序列的测试模型缺乏对调用服务内部约束的考虑。

上述方法均没有考虑到操作序列中测试数据之间的接口契约关系，以保障测试路径中的数据传递以及测试路径中驱动数据的合理有效，且现有操作序列的构建和测试数据的生成主要还是依赖于人工完成，过程比较繁琐耗时，存在明显不足。

* + 1. 本课题组研究现状

课题组在本课题相关领域进行了大量的前期研究工作与技术积累，提出了面向Web服务及其组合程序的测试方法[[[53]](#endnote-53),[[54]](#endnote-54),[[55]](#endnote-55),[[56]](#endnote-56)][29-32]。

特别的，课题组已经尝试将模型驱动的测试方法应用于业务流程执行语言（BPEL）[[[57]](#endnote-57)]服务组合程序，提出了一种**基于场景的BPEL测试用例自动生成技术**及支持工具原型的开发，该方法将BPEL流程转换为**UML活动图模型**，通过不同覆盖策略生成测试序列，利用约束求解工具求解出相对应的测试数据，产生面向场景的测试用例集。实验结果表明，模型驱动的测试方法可以有效监测服务组装程序中潜藏的故障[[[58]](#endnote-58),[[59]](#endnote-59)]。

但该法重点测试BPEL流程出现的常见错误，仅仅考虑针对控制流进行建模。

1. 行为模型驱动的服务组合程序测试用例生成技术(名字名字)

本节主要通过示例介绍行为模型驱动的服务组合程序测试用例生成技术的方法简述、关键技术及具体实现。

本文从服务调用视角对Web服务的行为进行测试，在事件模型中考虑服务调用上下文环境，提出了一种行为模型驱动的服务组合程序测试用例生成技术，开发相应支持工具，辅助所提技术的测试用例自动化生成。图3-1所示为行为模型驱动的服务组合程序测试用例生成技术的框架图，该框架主要包含五个部分，分别为解析扩展WSDL文档、服务行为模型图建立、测试序列生成、测试用例生成及测试执行：

1. EX-WSDL Paring：解析服务开发商提供的扩展WSDL、获取Web服务提供的操作及其约束(Constraint Result，用于行为模型生成)、调用操作的Soap消息框架(SoapEnv，用于测试数据填充)及XML结构定义文档(XSD，用于消息校验)；
2. Behavior Model Construction：针对提取出的约束文档(Constraint Result)生成该服务的行为模型(Model)；
3. Test Path Generation：针对该行为模型使用定义的覆盖准则，生成测试序列(Test Seqs)；
4. Test Cases Generation：根据测试序列，从决策表中找到符合该序列的执行约束，使用z3求解器求解出测试数据，填充到Soap消息中形成可执行的测试用例(Test Suite)；
5. Test Execution：集成SoapUI模拟客户端，执行测试用例，生成测试报告(Test Report)；

图3-1 技术框架图

因此本课题重点研究以下四个问题：

1. 服务行为约束的定义与描述
2. Web服务行为模型的建立
3. 基于建立模型的测试序列、测试数据与测试用例的自动生成
4. 测试执行与结果判定

接下来将分节介绍解决上述问题的关键技术及其具体实现。

* 1. WSDL文档行为约束扩展（2小节 包含Web服务行为约束的定义与描述，如何将约束添加到WSDL中去，方法示例）

由于服务使用者只能依据规格说明访问服务。然而服务规格说明仅仅包含了服务接口说明（调用的操作包含哪些数据以及数据格式要求），但是仅描述这些结构化的语法信息是不够的，参与组合的服务通过接口中的多个操作提供服务，这些操作潜藏的各种**数据**与**控制流**方面的约束往往是业务处理的关键。控制流约束表示操作之间的执行顺序约束，例如电子支付系统中“转账”操作依赖于“建立账户”这样的操作之后。数据流约束表示触发某个特定操作所需的特定数据状态，根据上下文数据决定是否执行相应的操作，例如，ATM转账业务中，当转账的金额不大于卡内余额时可触发转账操作。同时对于同一操作内部来说，输入数据范围的不同可能导致该操作的内部执行逻辑的不同。然而这类约束通常采用自然语言进行描述，因此需要考虑如何形式化的刻画与表达服务行为约束。

本节将介绍支持的约束类型的定义和应用场景及如何扩展WSDL支持约束的表达

* + 1. 行为约束类型定义与描述

通过阅读文献研究总结了如下六个Web服务提供操作自身及相互之间可能存在的约束：**时效约束**[33]、**序列约束**[27,34,42,43,44,[[60]](#endnote-60)]、**调用约束**[26]、**参数范围约束**[34,60,[[61]](#endnote-61),[[62]](#endnote-62)]、**参数关系约束**[7,61]、**区域约束**[33]。如图3-2所示，按照约束所在层次分为：**服务层**次约束、**操作层**次约束，其中操作层约束又分为**数据流**与**控制流**约束。



图3-2 服务约束定义与分类

本文使用巴克斯范式(Backus Normal Form)表述约束模型：

|  |
| --- |
| <Constrain> ::= <ConName><ConContent>|<ValueType><ValueRestriction>  <ConName> ::= eTime|ipRegion|preOp|Iteration|invokeOp|paraRelation  <ConContent> ::= date|<ipAdress>|<>|true|false|  <ValueType> ::= string|boolean|int|float|double  <ValueRestriction> ::= enumeration|maxExclusive|minExclusive|maxInclusive|minInclusive|pattern |

为准确描述约束类型的定义，我们使用JSON[[[63]](#endnote-63)]键值对描述服务行为。

* + 1. 扩展WSDL支持行为约束的表达

Web服务约束定义完成后，我们考虑如何在服务描述中存储包含约束的服务行为，即如何支持服务行为约束的表达。

现有两种方法向Web服务描述中添加行为信息，一是扩大现有WSDL协议，另一种是实现一个能够表达行为约束的新的协议。考虑到当前的WSDL作为服务描述的标准规范广泛应用于Web服务描述且新协议没有统一的标准，本文采取第一种方法，通过扩展WSDL支持服务行为约束表达。

现有研究大部分依赖XML可扩展标记，在WSDL中定义新的标签及命名空间，由于新定义标签众多且没有统一标准，本文考虑使用WSDL标准中包含的<documentation>元素支持服务行为约束表达。如图3-3 a)所示，WSDL[2]使用可选的<documentation>元素，作为服务开发和使用人员可读的文件的容器，元素的内容是任意的文本，且<documentation>元素被允许嵌套在任何其他WSDL标签元素中。<documentation>标签作为WSDL标准中的元素，出于其灵活性与广泛使用，我们使用<documentation>标签作为服务行为约束表达方式。



图3-3 WSDL扩展前后元素对比

我们针对标准WSDL服务层与操作层进行扩展，将<service>与<operation>标签内的可选元素<documentation>定为必选元素，如图3-3 b)所示，在其中添加服务及操作的相关行为约束。一个扩展后的WSDL文档如下所示：（例子例子）

* 1. 基于扩展WSDL文档的Web服务行为模型生成方法（一大节 + 方法示例）

定义了服务约束并扩展WSDL文档后，考虑基于服务行为的形式化描述模型的建立。本课题采用**事件序列图(ESG)**[[[64]](#endnote-64)]对服务行为进行**行为描述建模**，重点关注**服务进行了何种操作**，**操作的数据状态**及**操作之间的依赖**。事件序列图提供一种简洁、易于理解的形式化表达手段，由节点及边组成，节点代表触发的事件，边代表事件的转移，同时具有较强的理论技术，能够表达数据约束及控制依赖，有如下定义[32]。

1. 事件序列图(ESG)定义为2元组ESG =<V, E>。V表示ESG中全部（有限）节点集合，代表用户输入或系统响应的事件且V≠∅；Ξ⊆V，代表ESG中有限入口节点集合；Γ⊆V，代表ESG中有限出口节点集合；E表示ESG中的全部（有限）边集合，代表事件的有向转移——E⊆V×V；
2. 事件序列(ES)：对于一个序列<>，如果，则称该序列是事件序列；
3. 完整事件序列(CES)：事件序列的起点为入口节点且结束节点为出口节点，称该事件序列为完整事件序列。

我们对事件序列图进行扩展，定义Web服务行为模型（WSBM），Web服务行为模型定义为2元组WSBM=<N,D,V, E>。

1. N标识该模型名称；
2. D标识该模型有效期限，**从时效约束提取**；
3. V=<N,I,C,B,A,T,>标识WSBM中全部（有限）节点集合，其中N表示节点名称；I表示节点ID编号；C表示节点约束；B表示节点前继节点集合；A表示节点后续节点集合；T表示节点类型；模型图中定义了不同类型的节点及约束，节点类型包括：**开始节点（Start）、初始节点（Init）、结束节点（End）、请求节点（Req）、响应节点（Res）**，每种类型的节点及其具体含义如表3-1所示；请求节点的约束类型包括：**参数范围约束、参数关系约束、区域约束、序列约束、调用约束；**响应节点约束则要根据服务提供商提供的决策表来提取；
4. E=<N,TO,FR>是WSBM中的全部（有限）边集合。代表事件的有向转移——E⊆V×V。其中N表示边名称；TO表示边的初始点；FR表示边的终端点；
5. 事件序列(ES)：对于一个序列<>，如果，则称该序列是事件序列；
6. 完整事件序列(CES)：事件序列的起点为入口节点且结束节点为出口节点，称该事件序列为完整事件序列。

表3-1 Web服务行为模型节点类型定义

|  |  |
| --- | --- |
| 节点类型 | 节点描述 |
| Start | 表示Web服务行为模型入口 |
| Init | 表示流程初始化 |
| End | 表示Web服务行为模型出口 |
| Req | 表示服务请求事件 |
| Res | 表示服务响应事件 |

服务行为模型的UML类图如图3-4所示，



图3- 4 服务行为模型的UML类图

其中

|  |  |
| --- | --- |
| Algorithm 1 解析EX-WSDL生成服务行为模型算法 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | 初始化一个行为模型g（Nodes=Ø、Edges=Ø），将其名称属性设置为EX-WSDL URI中解析出的服务名称  将其有效时间设置为服务时效约束中的eTime  向Nodes中添加一个Start节点及Init节点，将Start节点添加进Init节点的前继节点，Init节点添加进Start节点的后续节点  解析EX-WSDL，获取其全部操作集合OpSet  **for** each op = element in OpSet **do**  创建opReq节点，设置节点类型为input，解析op操作的paraRange、ipRegion添加进opReq节点约束属性中  创建opRes\_succ节点，设置节点类型为output，设opReq为opRes\_succ前继节点；opRes\_succ为opReq后续节点  创建opRes\_fail节点，设置节点类型为output，设opReq为opRes\_fail前继节点；opRes\_fail为opReq后续节点  向Nodes中添加opReq及opRes节点  **if** opIteration == true **then**  设置opRes\_succ为自己的前继及后续节点  **end if**  **if** op preOp == null **then**  opReq添加进Init的后续节点，Init添加进opReq前继节点  **else**  解析preOp约束，将有前后关系的节点进行关联  **end if**  **end for**  **for** each node in Nodes **do**  **for** afterNode in node. afterNodes **do**  Add edge to Edges  **end for**  **end for** |

* 1. 基于行为模型的测试用例生成技术

理论上，服务应该是无状态的（即服务提供的操作可以任意顺序调用）。然而实践中，这些操作则存在一定的数据流、控制流约束关系（3.1小节中提及）。为了验证服务行为的正确性和可靠性，必须充分的测试所有服务行为相关的数据流和控制流约束信息。每个从服务组合流程开始到结束的路径称为一条测试序列，在给定的行为模型基础上，定义各种覆盖准则，开发相应的遍历算法，即可获取所需测试序列以及程序执行该条序列的数据约束，可通过约束求解获取该测试序列对应的测试数据，最后将数据进行封装生成测试用例，其过程如图3-5所示。



图3-5 测试用例生成过程

因此我们重点考虑如下三个问题：

1. **如何从给定行为模型中生成测试序列；**
2. **如何根据测试序列求解相对应的测试数据；**
3. **如何将测试数据与测试序列结合，生成可执行的测试用例。**
   * 1. 基于行为模型的测试序列生成

行为模型建立完毕后，考虑如何从Web服务行为模型中依据覆盖算法导出目标测试序列，在Web行为模型中从开始节点到结束节点的一条路径是一个测试序列，对于不同测试角度而言，关注的测试序列不同，因此我们提出以下的覆盖准则[[[65]](#endnote-65)]来决定模型中哪些元素需要进行测试。

1. Request-Node Covering（请求节点覆盖）：要求设计足够多的测试用例，使得Web服务行为模型图中的每个请求节点至少被覆盖一次；
2. Response-Node Covering（响应节点覆盖）：要求设计足够多的测试用例，使得Web服务行为模型图中的每个响应节点至少被覆盖一次；
3. Edge covering (边覆盖)：要求设计足够多的测试用例，使得Web服务行为模型图中的每条边至少被覆盖一次；
4. State coverage（状态覆盖）：该覆盖准则结合前三条覆盖准则一起使用，前面所述三条覆盖准则，并未考虑覆盖到该消息节点或执行边时的服务状态，状态覆盖要求设计足够多的测试用例，使得覆盖Web服务行为模型图中目标元素的同时考虑其内部状态的覆盖。

本课题集成开源模型测试工具GraphWalker[15]实现测试序列的自动生成，GraphWalker提供了一种Graph Model Language以及多种图模型遍历策略。下面本文将介绍所使用的图模型遍历策略以及如何将定义的覆盖准则转换为相对应的遍历策略。

GraphWalker遍历策略由路径生成器(Path Generators)与停止条件(Stop Conditions)组成，其中路径生成器是一种决定如何遍历模型的算法，生成器提供了随机算法、起始点算法及最短路径算法；停止条件是决定何时停止该遍历过程（如模型元素覆盖率达到设定比例或特定元素被覆盖），停止条件有节点覆盖比例、边覆盖比例、特定节点覆盖、特定边覆盖、执行时间等。

本文使用**起始点算法**及**特定节点覆盖**与**特定边覆盖**停止条件进行行为模型的遍历。该算法其表达方式为：

**a\_star**( a stop condition that names a **vertex** or an **edge** )

起始点算法将从模型起始节点进行遍历，生成覆盖特定节点或边的最短路径，因此针对本文定义的覆盖准则，仅进行如下转换：

* + 1. 基于约束求解策略的测试数据自动生成

本节介绍基于约束求解的测试数据自动生成，包括约束条件表达式的提取、约束求解工具Z3自动生成可行解。

在前面的论述中，我们知道测试用例生成包括测试序列的生成、测试数据的生成。上一小节，已经根据不同覆盖策略生成了测试序列，生成测试序列后，考虑如何根据测试序列，生成满足该序列的测试数据。

传统测试方法中，测试数据的获取主要依靠人工分析来完成，不仅繁琐而且耗时，并且只覆盖部分所需的数据。为了保证生成测试用例的完备与有效性并且提高测试质量，本文针对每条测试序列自动生成测试数据。

现有测试序列生成测试数据主要有以下三种方法[42,43,44]：

1) 基于符号运算的静态方法

符号运算是指不运行程序，人工分析路径上的有关变量和谓词，对路径上的变量及条件判断变量用符号表达式进行替换。其对简单的线性路径很适用，但是即使是对线性问题也会遇到困难：(1)循环的运算；(2)过程调用；(3)数组和指针的处理；(4)不可达路径问题。

2) 基于遗传算法的程序直接执行

遗传算法通过模拟自然界生物进化的思想，以遗传算法和自然选择为理论，从而实现搜索优化。遗传算法在解决大空间、非线性、全局优化等复杂问题时显示出其优越性，但其实现却面临着一系列的问题，例如参数编码和解码、适应度函数的选择、遗传算法自身参数设定等。

3) 基于约束求解策略的方法

分析对应路径的所有约束条件表达式，通过求解得到测试用例。约束求解包括如下两个步骤，一是提取约束条件，即从被测试程序中提取出对应路径的所有约束条件表达式；二是求解约束，即采用正确的算法和策略，求出约束条件表达式的若干可行解。

本文将已有的测试序列，进一步采用**约束求解技术**（Constraint Solver）在测试序列基础上产生所需测试数据。在约束求解步骤中综合使用线性约束求解策略与迭代求解策略，线性约束求解策略求解出所有的解，然后用迭代求解的策略选出满足约束的解，从而得出满足每条测试序列的测试数据。该方法处理流程如图3-6所示。



图3-6 测试数据生成流程

约束求解包括如下两个步骤，一是**提取约束条件**，即从行为模型中提取出对应序列的所有约束条件表达式；二是**求解约束**，即采用正确的算法和策略，求出约束条件表达式的可行解。

下面重点介绍测试数据生成流程中的关键技术：**约束求解工具Z3自动生成可行解**、**约束条件表达式的提取与转换**。

1. **约束求解工具Z3自动生成可行解**

Z3提供了很多的底层接口（**SMT-LIB**、C/C++、.NET、JAVA）来实现约束条件表达式的求解，并且拥有自己的语法格式。本文使用SMT-LIB语言编写Z3求解脚本。

Z3 SMT-LIB求解脚本的输入格式有**变量定义、运算**及**模型求解**三个部分。

变量定义部分使用**declare-const**命令定义需要求解的变量：Z3内置整数(Int)、实数(Real)、布尔数（Bool）、字符串（String)及常量支持，declare-const命令语法为：**declare-const 变量名 变量类型**。

运算部分使用**assert**命令添加约束到Z3内部堆栈，用于后续求解：使用巴克斯范式表示assert命令规则如下所示：

|  |
| --- |
| assert <term>  <term> ::= <symbol> <op> <op>  <symbol> ::= > | < | >= | <= | = | + | - | \* | / | div | mod | rem  <op> ::= <var> | <vaule> | <term>  <var> ::= Int | Real | Bool | String  <vaule> ::= the value that conforms to the variable type |

模型求解部分使用**get-model**命令获取约束解，也可以使用**get-value (var)**命令获取特定变量var的求解值。

如图3-7所示为Z3求解脚本示例及其求解结果，约束要求为：(a+b)<=5 && a>=2 && d=”hello”，Z3判定约束表达式可求解后（使用chack-sat命令判断）即输出所求变量可行解：a=2;b=0;c=false;d=”hello”。



图3-7 Z3求解脚本示例及其求解结果

1. **约束条件表达式的提取**
   * 1. 测试用例集生成

生成测试数据后，需要考虑如何将测试数据与测试序列结合，生成可执行的测试用例。

本文使用XML Format定义测试用例，其语法如图3-8所示。其中根节点为<Sequence>标签，用来标识测试用例；<Operation>标签标识被调用操作，name属性代表被调用操作名称，id属性代表操作被调用次数，<Sequence>标签中可包含多个<Operation>标签，<Operation>标签的前后顺序即为服务调用的先后顺序；<Operation>标签内包含调用该服务需要发送的SOAP消息，<Body>标签内包含消息所需测试数据。

|  |
| --- |
| <Sequence> |
| <Operation name="opName" id="idNumber">\*  <soapenv:Envelope xmlns:soapenv="envelope url">  <soapenv:Header/>  <soapenv:Body>  <-- Data to be filled -->  </soapenv:Body>  </soapenv:Envelope>  </Operation> |
| </Sequence > |

图3-8 测试用例框架

定义完测试用例格式后，我们考虑如何将测试数据与测试用例进行结合。在前面的技术框架图论述中我们提出将使用z3求解器求解出测试数据填充到扩展WSDL分析阶段生成的Soap消息中与生成的测试用例合成可执行的测试用例。如图3-9所示，测试用例从**数据层**、**消息层**、**序列层**三个层次进行生成：

1. 测试数据基于约束求解析生成；
2. 将测试数据填充到扩展WSDL分析阶段生成的相应SOAP消息框架中，生成单个操作测试消息；
3. 按照测试序列，将测试消息进行组合后，生成最终的测试用例。



图3-9 测试用例生成框架

* 1. 测试执行与结果判定

|  |  |
| --- | --- |
| Algorithm 2 正面测试序列生成算法 | |
| 1  2  3  4  5  6  7  8  9  10  11  12 | 初始化初始测试序列集initTsmap=Ø，去冗余后测试序列集tsm=Ø  遍历行为模型图获取需要覆盖的元素集合eleCoverSet  **for** element in eleCoverSet **do**  生成覆盖element测试序列，添加进initTsmap集合  **end for**  **while** eleCoverSet != Ø **do**  取initTsmap中最长序列ts  找到ts覆盖的所有element集合eleTCoverSet  eleCoverSet = eleCoverSet – eleTCoverSet  initTsmap = initTsmap – ts  tsm = tsm + ts  **end while** |

|  |  |
| --- | --- |
| Algorithm 2 负面测试序列生成算法 | |
| 1  2  3  4  5  6  7  8  9  10  11 | 遍历行为模型图中类型为input的节点inputNodeSet  **for** node in inputNodeSet **do**  **if** node.Iteration == false **then**  生成满足重复调用node消息的测试序列nts  Add nts to tsm  **end if**  **if** node.preOp != null **then**  生成直接调用node的测试序列nts  Add nts to tsm  **end if**  **end for** |

本章介绍行为模型驱动的Web服务测试用例生成技术的基本原理以及一般过程，并用实例介绍该技术的使用。

* 1. 基于包含关系的变异测试优化技术MuSR
     1. 方法原理

通过研究BPEL程序的变异算子操作规则，本文发现不同变异算子指导产生的变异体可能具有相似或相同的错误。在变异测试中，识别并移除出这些具有相似或相同的错误的变异体，可以降低变异体数量同时不影响测试效果。基于以上想法，本文提出基于包含关系的变异测试优化技术，简称为MuSR技术，是一种通过移除具有包含关系的变异算子进行变异测试的优化技术。其中，包含关系定义如下：

假设存在两个变异算子*OA*和*OB*，对于给定的程序分别生成*MA={M1A , M2A ,…, MjA }*，*MA={M1A , M2A ,…, MjA }*变异体集合，若存在规则 杀死 *MkA*，*MlB*也一定被 *tc* 测试用例杀死，则说*OA*包含*OB*，记为*OA←OB*。



满足包含关系规则，说明*MlB*与*MkA*的错误行为是等价的。那么，*OB*产生的变异体可以被*OA*产生的变异体所替代和精简。

对BPEL程序的34种变异算子规则和产生的变异体进行研究，共发现如下7组具有包含关系的变异算子组合：*AEL←AIE、ASF←ASI、ERR←EIN、AEL←CFA、EIU←EAN、CDC←CDE*和*CDC←CCO*。接下来，以实例阐述这7组变异算子组合的包含关系。

1. *AEL*变异算子描述的删除一个活动的操作。*AIE*变异算子描述的是移除*if*活动中的*else if*活动或*else*活动的操作。显然，*AEL*变异算子可以移除BPEL程序中的任意活动，其中也包括*if*活动中的*else if*活动或*else*活动，所产生的变异体等价于*AIE*产生的变异体，如表3-1所示。可知，*AIE*产生的变异体完全被包含在*AEL*产生的变异体集合中。
2. *ASI*变异算子定义交换*sequence*活动内的两个子活动顺序的操作。*ASF*描述的是用*flow*活动替换*sequence*活动的操作。*flow*表示支持并发的活动。因为将*sequence*活动替换为*flow*活动后，其子活动执行顺序将不确定，此举与*ASI*算子行为相同，如表3-2所示，所以，*ASF←ASI*。

表3-3 AIE与AEL变异算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <if name="If1">  <condition>......</condition>  <sequence>......</sequence>  <else>......</else>  </if> | 无 |
| AIE | <if name="If1">  <condition>......</condition>  <sequence>......</sequence>  </if> | 删除if活动中的else活动 |
| AEL | <if name="If1">  <condition>......</condition>  <sequence>......</sequence>  </if> | 删除else活动 |

表3-4 ASI与ASF变异算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <sequence name="Sequence1">  <invoke ......> ......</invoke>  <assign name="Assign">......</assign>  </sequence> | 无 |
| ASI | <sequence name="Sequence1">  <assign name="Assign">......</assign>  <invoke ......>...... </invoke>  </sequence> | 交换assign与invoke活动 |
| ASF | <flow name="Sequence1">  <invoke ......>...... </invoke>  <assign name="Assign">......</assign>  </flow> | 替换sequence为flow活动 |

1. *EIN*变异算子是在逻辑表达式前插入*not*语句，就是对逻辑表达式的值取反的操作。*ERR*变异算子是将关系运算符用另一个相同类型运算符替换的操作。若*ERR*将逻辑表达式中的“≥”替换为“<”，这与逻辑表达式取反是同样的错误，如表3-3所示。可知，*ERR←EIN*。
2. *CFA*变异算子是移除流程中的一个活动，用*exit*活动替换的操作。*exit*活动用于立即结束业务程序实例。*AEL*变异算子定义为移除一个活动的操作。两种变异算子均包括移除任一活动操作，所造成的结果相似，如表3-4实例所示。此外，*CFA*是移除活动并结束业务程序，产生的变异体相比AEL的变异体更容易被杀死。
3. *EAN*变异算子定义为替换表达式为其绝对值的负值的操作。*EIU*变异算子是对算数表达式取负的操作。如表3-5所示，该实例中这两个变异算子产生的变异体相同。

表3-5 ERR与EIN算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <condition>  ($warehouseAmessage.WarehouseAResponse &gt;= $input.amount)  </condition> | 无 |
| EIN | <condition>  not(($warehouseAmessage.WarehouseAResponse&gt;=$input.amount))  </condition> | 条件表达式前加NOT |
| ERR | <condition>  ($warehouseAmessage.WarehouseAResponse &lt; $input.amount)  </condition> | 替换&gt;= 为&lt; |

表3-6 CFA与AEL变异算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <sequence name="Sequence1">  <invoke......> ......</invoke>  <assign name="Assign">......</assign>  </sequence> | 无 |
| CFA | <sequence name="Sequence1">  <exit ......> ......</exit>  <assign name="Assign">......</assign>  </sequence> | 用exit活动替换invoke活动 |
| AEL | <sequence name="Sequence1">  <assign name="Assign">......</assign>  </sequence> | 移除invoke活动 |

表3-7 EAN与EIU算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <transitionCondition>  ($decryptResponse.creditInformationMessage/amount &lt;= 1000.0)  </transitionCondition> | 无 |
| EAN | <transitionCondition>  ($decryptResponse.creditInformationMessage/amount &lt;= -(1000.0 \* ((1000.0 &gt;= 0.0) - (1000.0 &lt; 0.0))))  </transitionCondition> | 替换1000为其绝对值并取负 |
| EIU | <transitionCondition>  ($decryptResponse.creditInformationMessage/amount &lt;= -1000.0)  </transitionCondition> | 用-1000替换1000 |

1. *CDC*变异算子是应用判定/条件覆盖准则到程序中的操作。判定/条件覆盖准则指判定中的每个条件都取到各种可能的值，并使每个分支也要取到各种可能的取值。*CDE*变异算子是应用判定覆盖准则到程序中的操作。判定覆盖指的是每个分支至少执行一次。在表3-6给出的实例中，这两种变异算子产生的变异体相同。此外，*CDC*变异算子所能产生的变异体相比*CDE*的变异体更加全面。可知，*CDC←CDE*。
2. *CDC*变异算子是应用判定/条件覆盖准则到程序中的操作。*CCO*变异算子是应用条件覆盖准则到程序中的操作。条件覆盖准则是指使程序里每个判断中每个条件的可能取值至少执行一次。在表3-7给出的实例中，这两种变异算子产生的变异体具有相同错误行为；并且*CDC*变异算子所能产生的变异体相比*CCO*的变异体更加全面，可知，*CDC←CCO*。

表3-8 CDE与CDC变异算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <condition>  $quantity.CheckQuantity&lt; $commodity.amount  </condition> | 无 |
| CDE | <condition>true()</condition> | 将条件表达式取真 |
| CDC | <condition>true()</condition> | 将条件表达式取真 |

表3-9 CCO与CDC变异算子应用实例对比

|  |  |  |
| --- | --- | --- |
| **变异算子** | **程序片段** | **操作** |
| 原始程序 | <condition>  $quantity.CheckQuantity&lt; $commodity.amount  </condition> | 无 |
| CCO | <condition>true()</condition> | 将条件表达式取真 |
| CDC | <condition>true()</condition> | 将条件表达式取真 |

* + 1. 一般过程

MuSR技术的一般过程是：首先，运用包含关系规则，识别出具有包含关系的变异算子；然后，从变异算子集合中删除被包含的变异算子，获得精简的变异算子子集；最后，使用变异算子子集指导产生变异体，进行变异测试。

* 1. 基于二阶变异体的变异测试优化技术MuSOM
     1. 方法原理

对于变异测试开销大的问题，存在一种高阶变异体优化技术。该技术是由高阶变异体替代一阶变异体进行测试的优化技术。其中，一阶变异体指对原始程序应用一个变异算子且植入一处错误而生成的变异体；高阶变异体指对原始程序植入*M（M≥2）*处错误而生成的变异体。一个高阶变异体可以看成由多个一阶变异体组成。高阶变异体的优化技术的提出基于两个推测[11]：1）执行一次*M*阶变异体等同于执行*M*个一阶变异体；2）高阶变异体比一阶变异体产生等价变异体的概率小。基于这两个推测，可以看出高阶变异体优化技术主要是通过减少待执行变异体数目和等价变异体识别开销来降低测试开销。

指导生成高阶变异体的方法有很多，这里讨论一种基于*LastToFirst* 算法[39]来生成二阶变异体的方法。*LastToFirst* 算法描述按照一定顺序组合两个一阶变异体，产生二阶变异体的过程，其原理如图3-1所示。其中，*P*指待测程序，*M1*至*Mn*代表原始程序的一阶变异体，*Mk,k+1*代表通过组合两个一阶变异体而生成的二阶变异体。本文将*LastToFirst* 算法应用到面向BPEL程序的变异测试的中，提出面向BPEL程序的二阶变异体优化技术，简称为MuSOM。



图3-10 *LastToFirst* 算法生成二阶变异体过程

* + 1. 一般过程

MuSOM描述的二阶变异体生成步骤如下：

1. 为BPEL程序*P*生成一阶变异体；按照一阶变异体产生的顺序，给变异体命名并编号为*1*至*n*，*n*为一阶变异体的总数；得到的变异体的集合表示为*M1o={M1,M2,...,Mk, Mk+1,...,Mn-1,Mn}*。
2. 将*M1o*中的一阶变异体组合生成二阶变异体，组合方法为：将编号为*1*与编号为*n*的两个一阶变异体组合，生成二阶变异体，记为*M1,n*；将编号为*2*与编号为*n-1*的两个一阶变异体组合，生成二阶变异体，记为*M2,n-1*；以此类推，使得每个一阶变异体都用来生成二阶变异体。
3. 当*n*为奇数时，除了一阶变异体M(n+1)/2，其余的一阶变异体都两两组合产生了二阶变异体。这种情况下，规定将M(n+1)/2与M(n+1)/2+1变异体进行组合，来生成二阶变异体。
4. 最终，得到二阶变异体集合为*M2o={M1,n,M2,n-1,...,Mk,k+1}*。
   * 1. 方法示例

为了方便理解MuSOM技术，使用SupplyChain[10]实例进行演示。SupplyChain是一个BPEL实例，同时也在第5部分用于实例验证。SupplyChain实例共生成34个一阶变异体。采取MuSOM方法，为该实例生成二阶变异体。图3-2展示一个二阶变异体生成过程。其中，*CDE\_1*代表生成的第一个一阶变异体的部分程序代码。该变异体使用了*CDE*变异算子，应用判断准则到程序中，将条件表达式用*true()*替换。图中对变异的位置用加粗放大字体进行标识。*ASI\_4*代表生成的最后一个一阶变异体部分程序代码。该变异体应用*ASI*变异算子，交换了*reply*活动与*if*活动的位置。通过读取这两个变异体所植入的错误，同时应用到原始程序*P*中，得到一个二阶变异体，命名为*CDE1\_ASI4*。



图3-11 SupplyChain的二阶变异体生成示例

* 1. 基于变异算子优先级的变异测试优化技术MuPri
     1. 方法原理

1. **基本原则**

在变异测试中，有的变异体可以被绝大多数测试用例“杀死”，而有的变异体只能被特殊的测试用例检测出来。若一些较难被测杀的变异体都能被测试用例“杀死”，那么理论上该测试用例也能“杀死”那些容易被测杀的变异体。基于以上内容，我们从变异体执行顺序的角度来考虑减少测试开销，优先使用这些可以产生较难被测杀变异体的变异算子，理论上可以生成更难被测杀的变异体集合，这些变异体可以更好的衡量测试用例有效性和完备性。

1. **相关规则定义**

为了衡量变异算子产生变异体被测杀的难易程度，我们引入一个变异算子质量度量指标。在提出该指标之前，约定*P*指待测的BPEL程序；*TS*为有效测试用例集，*TS={t1,t2, …,ti,…,tn}*，其中*ti* 为测试用例集中第*i*个测试用例，*n*为测试用例集的用例总数；*MO*表示BPEL语言的变异算子集合，*MO={O1,O2,…Oi,…,O34}*，其中*Oi*表示变异算子列表中第*i*个变异算子。

变异算子质量的定义从故障检测率推导而来，**故障检测率**（Fault Detection Rate，FDR）[4]定义为测试用例“杀死”变异体的比例，广泛用来衡量用例的故障检测能力。FDR计算公式为：

（3-1）



其中，*Mji* 表示*Oi* 变异算子的第*j*个变异体*，NC(Mji , TS)*表示杀死变异体*Mji* 的测试用例数目。FDR值越大，说明测试用例集*TS*杀死该变异体的几率越大，该变异体越容易被杀死。从而，将**变异算子质量**定义为：

 （3-2）

其中，*NOi* 表示*Oi* 变异算子产生的非等价变异体总数。通过FDR的定义可知，*Qo*的值越大，表明越少的测试用例将该类变异算子产生的变异体杀死，反映出该类变异体的质量较好。

* + 1. 一般过程

本文提出一种基于变异算子优先级的变异测试优化技术，简称为MuPri。MuPri是通过衡量变异算子质量，为变异算子分配测试优先级的一种优化技术。MuPri技术基本过程为：首先对BPEL程序进行变异测试；然后根据测试结果，计算变异算子的质量，按照质量由高到底的顺序为其排序，质量好的变异算子在变异测试中分配较高的优先级，质量差的变异算子分配较低的优先级；按照变异算子优先级顺序指导生成变异体集合，这些变异体则是按照被测试用例“杀死”由难到易的顺序生成。通过这个顺序来选择测试用例，可以优先选出能将较难“杀死”的变异体检测出的测试用例，从而可以达到测试用例排序的效果，并可以更快捷的衡量测试用例的有效性和完备性。所以，MuPri技术不仅可以给测试用例集合排序，得到检错效率更高测试用例集，而且可以依据变异算子的优先级，优先使用质量好的变异算子生成变异体，增强变异体集质量并减少测试开销。其中，MuPri技术为测试用例排序的一般过程描述如下：

1. 选取实例程序*P*，将其可应用的变异算子加入到集合*O*中；
2. 根据变异算子的优先级排名，按照排名从小到大的顺序，依次取出变异算子加入到*OP*队列中，得到*OP=[OP1,…,OPZ,…,OPn]*；
3. 令*z*=1；
4. 取出*OPZ*变异算子，生成其一阶变异体集合*MOZ(P)*；
5. 使用测试用例*TS*执行*MOZ(P)*中的变异体，其中，*TS*表示具有顺序的用例队列。根据执行结果，从*MOZ(P)*中删除被*TS*“杀死”的变异体。判断*MOZ(P)*是否为空，如果是则执行第七步；否则执行第六步；
6. 添加一个测试用例到*TS* 中并返回上一步；
7. 判断*z*是否等于*n*，如果是，则输出测试用例*TS*；否则令*z*=*z*+1，返回第四步。
   * 1. 方法示例

采用SupplyChain实例演示MuPri技术为测试用例排序过程。首先对该实例进行变异测试，统计变异算子质量，得到变异算子顺序为“*CDE->CCO->CDC->ERR->AIE->ASF->AEL->CFA->EIN->ASI->ACI*”。按照3.3.2节提出的一般过程，对SupplyChain实例的测试用例进行排序，结果如表3-8所示。表中的对勾表示该测试用例是测试用例集*TS*中，第一个将该变异体“杀死”的用例。因此，这个实例根据MuPri技术得到测试用例集的顺序是“*T1->T2->T3*”。

表3-10 SupplyChain变异体执行测试用例*TS*结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Rank** | **Mutant** | | **Test Case** | | |
| T1 | T2 | T3 |
| 1 | CDE | 1 | √ |  |  |
| 2 |  | √ |  |
| 2 | CCO | 1 | √ |  |  |
| 2 |  | √ |  |
| 3 | CDC | 1 | √ |  |  |
| 2 |  | √ |  |
| 4 | ERR | 1 | √ |  |  |
| 2 |  |  | √ |
| 3 | √ |  |  |
| 4 |  | √ |  |
| 5 | √ |  |  |
| 5 | AIE | 1 |  | √ |  |
| 6 | ASF | 1 | √ |  |  |
| 2 | √ |  |  |
| 7 | AEL | 1 | √ |  |  |
| 2 | √ |  |  |
| 3 | √ |  |  |
| 4 | √ |  |  |
| 5 | √ |  |  |
| 6 |  | √ |  |
| 7 | √ |  |  |
| 8 | CFA | 1 | √ |  |  |
| 2 | √ |  |  |
| 3 | √ |  |  |
| 4 | √ |  |  |
| 5 | √ |  |  |
| 6 |  | √ |  |
| 7 | √ |  |  |
| 9 | EIN | 1 | √ |  |  |
| 10 | ASI | 1 | √ |  |  |
| 2 | √ |  |  |
| 3 | √ |  |  |
| 4 | √ |  |  |
| 11 | ACI | 1 | √ |  |  |

* 1. 小结

本章结合BPEL程序基本特点和变异测试基本原理，从变异算子之间的包含关系、变异算子的优先级和二阶变异技术三个方面进行研究，提出了三种变异测试优化技术；阐述了优化技术的基本原理，一般过程，并利用SupplyChian程序实例来演示了MuSOM和MuPri技术的具体过程。

1. 行为模型驱动的Web服务测试用例生成工具设计与实现

本章介绍行为模型驱动的Web服务测试用例生成支持工具的设计与实现，包括需求分析、工具设计与实现，最后用一个实例验证并演示工具。

* 1. 需求分析

为了便于使用本文提出的行为模型驱动的Web服务测试用例生成技术，本文开发了一个行为模型驱动的Web服务测试用例生成支持工具CBWSTC（A System for Web Service Test Case Generation）。CBWSTC支持扩展后WSDL解析、行为模型的生成、测试序列生成、测试用例生成、执行和测试结果验证。采用UML用例图对CBWSTC进行需求分析，如图4-1所示。



**图4-1 CBWSTC工具用例图**

下面将对图4.1进行各个用例的具体描述。

1. **待测程序选择：**

用户提供待测Web服务的扩展后WSDL访问地址，系统根据地址获取扩展后WSDL文档，并创建一系列所需工作目录。其包括如下两个子用例：

* **待测程序导入**：该用例用于待测服务的扩展后WSDL文档的导入，用户可提供该服务扩展后WSDL的URI或者本地WSDL文档绝对路径，工具将获取的文档导入到特定目录，以备后续相关操作。
* **工作目录生成：**工具根据用户选定的待测服务，生成后续操作需要的工作目录。生成的目录结构及其功能分别如图4.2、表4.1所示。

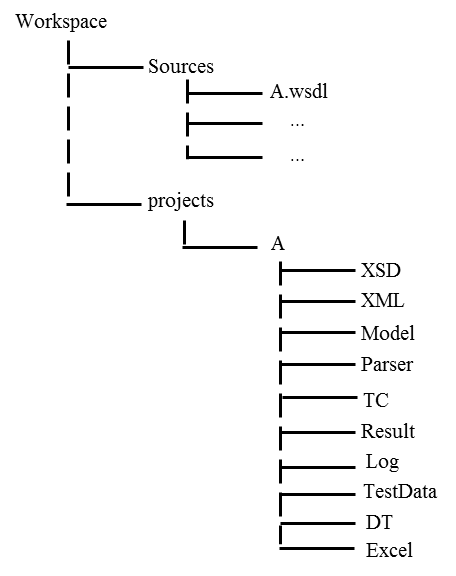


图4-2 工具目录结构

表4- 1 工具目录名称及其功能

|  |  |
| --- | --- |
| 目录名称 | 目录功能 |
| Source | 存储待测试服务的扩展后WSDL文档 |
| Projects | 待测服务工作目录 |
| XSD | 存储待测服务提供操作的输入输出接口描述 |
| XML | 存储待测服务提供操作的SOAP消息框架 |
| Model | 存储生成的行为模型 |
| Parser | 存储解析扩展后WSDL的约束信息 |
| Excel | 存储与服务有关的决策表 |
| DT | 存储决策表约束内容 |
| TestData | 存储生成的测试数据 |
| TC | 存储生成的测试用例 |
| Result | 存储测试结果 |
| Log | 存储测试日志 |

1. **行为模型生成：**

该用例解析扩展后的WSDL并将解析结果转换为服务的行为模型。其中包含如下三个子用例：

* **扩展后WSDL解析：**该用例通过解析扩展的WSDL，识别出该服务的约束描述、该服务提供的操作、操作的输入输出参数列表、操作调用消息框架以及操作的约束描述，存储解析结果。
* **服务约束提取：**该用例通过解析服务的约束描述及操作的约束描述，获得服务及操作具有的特定约束。
* **行为模型生成：**针对提取的服务及操作约束，根据设计的行为模型生成算法，采用事件序列图对Web服务进行行为模型的建立，获得待测Web服务的行为模型，使用Graphviz进行行为模型的可视化，并将行为模型转换为符合GraphWalker的图模型语言，方便后续测试用例生成。

1. **测试用例生成：**

生成行为模型后，根据覆盖准则遍历行为模型生成测试用例。其中包含如下三个子用例：

* **测试路径生成**：根据生成的行为模型，基于一定覆盖准则，获得测试路径。
* **测试数据生成：**针对每条测试路径中的操作以及操作输入变量的约束条件生成对应的测试数据。
* **测试用例生成：**根据服务操作调用框架，将测试数据与测试序列相结合，生成可执行的测试用例。

1. **测试执行：**

控制测试执行，从测试用例集检索测试用例、监视测试运行、搜集测试结果最终生成测试报告。包含如下三个子用例：

* **测试用例导入：**导入生成的测试用例。
* **测试用例执行：**在待测程序上运行测试用例，使用定义的一系列校验规则检测测试用例执行状态。
* **测试结果统计：**该用例通过对测试结果进行统计，给出执行通过与失败的测试信息，并生成测试报告。

1. **测试报告查看：**

测试者可查看测试结果报告，获取更加全面的测试信息，包含执行了哪些测试用例、测试用例执行结果、执行测试所花费的时间开销等信息。

* 1. CBWSTC设计与实现

针对需求分析，本节详细介绍CBWSTC工具的架构、各组件的设计及其实现。

* + 1. 系统架构

图4-3描述了CBWSTC工具的系统架构。选用橙色矩形框表示工具的基本组件，包括扩展WSDL解析组件，测试序列生成组件，测试用例生成组件及测试用例执行组件共四个组件。每个组件由多个模块构成，其中测试序列生成组件由行为模型生成及测试序列生成模块构成、测试用例生成组件由测试数据生成及测试脚本生成模块构成。各个组件的功能设计描述如下：



图4- 3 xxx工具系统架构图

1. **扩展的WSDL解析组件：**负责对待测程序的扩展后WSDL文件解析，生成相应行为约束文档（ParseResult），包括三个子模块。

* **扩展WSDL解析器**：该模块根据用户输入的扩展后WSDL文件路径，读入待解析扩展后WSDL文件，并进行解析。
* **XML文件读/写器：**负责生成待测服务操作的输入输出XML结构定义文件（XSD），为后续验证做准备。通过DOM4J技术读入扩展WSDL文件，将其解析为DOM树形结构的Document对象，对其中的**schema**节点进行查找，完成XSD文件的输出。
* **Soap消息生成器：**集成*soapUI*工具，调用其API生成服务操作Soap消息框架，便于后续测试数据填充。

1. **测试序列生成组件：**主要负责解析扩展WSDL解析器生成的行为约束文档，转换为行为模型图（ESG）；进一步对模型进行遍历，生成满足一定覆盖准则的测试路径。该组件包含两部分：
2. **行为模型生成组件**

* **行为模型生成器：**该模块通过解析扩展WSDL解析器生成的行为约束文档，根据行为模型生成算法，生成对应的行为模型（ESG）。
* **模型可视化：**该模块通过集成*Graphviz*，可视化展示行为模型。
* **行为模型转换器：**根据模型转换算法，将行为模型转换为*GraphWalker*支持的图模型文档（graphml），方便后续测试序列生成。

1. **测试序列生成组件**

* **合规测试序列生成器：**该模块根据用户选择覆盖准则，遍历graphml文档，生成满足特定覆盖准则的合规测试序列。
* **冲突测试序列生成器：**该模块根据冲突测试序列生成算法，遍历行为模型（ESG），生成冲突测试序列。

1. **测试用例生成组件：**该组件主要将测试序列求解出的执行该序列的测试数据，并将测试序列与数据进行合成生成测试用例，该组件包含两个部分：
2. **测试数据生成组件：**该组件主要负责提取测试序列约束条件表达式，集成开源工具*Z3*对提取出的条件表达式进行求解，生成满足约束条件的可行解。

* **约束提取器：**该模块主要针对每条测试序列，提取出执行该序列的约束条件，将其转换为符合Z3脚本语法的约束条件表达式。
* **约束求解器：**该模块集成开源工具Z3对提取出的约束条件表达式进行求解，生成满足约束条件的可行解。

1. **测试脚本生成组件：**该组件根据服务操作调用框架，将测试数据与测试序列相结合，生成可执行的测试用例。

* **数据填充器：**该模块主要将测试数据填充到相应的Soap消息框架中（消息框架已由解析器解析得到），生成Soap消息；
* **脚本生成器：**该模块根据测试序列中的操作调用顺序，将生成的Soap消息进行合成，生成可执行的测试用例。

1. **测试用例执行组件：**该组件负责执行测试用例并评估执行结果、生成测试报告，由如下三个模块组成：

* **用例执行器：**该模块依次读取测试用例生成组件生成的测试用例并解析、集成*soapUI*模拟客户端调用Web服务进行测试用例执行，保存执行过程中的输出结果。
* **结果评估器：**该模块监控用例执行过程，截取执行器执行过程中的请求响应信息、将截取信息与解析器组件生成的XSD文档进行校验，根据定义的校验准则，生成测试评估结果。
* **结果统计器：**该模块获取结果评估器生成的评估结果，对其进行统计分析并生成测试报告，包括（报告的信息）
  + 1. 工具实现

下面讨论支持工具xxx的实现。xxx核心功能的类图如图4-3所示，分为5大模块：

1. **BPEL程序解析Parser类**

该类主要对输入的BPEL程序使用DOM4J技术进行解析，找到可变异位置等信息。

* **Parser类：**用于解析BPEL文件的 document 对象，获取可变异位置信息列表。
* **ManageDataInfo类：**用于存储可变异位置等基本信息，包括变异算子名称、变异节点等。

1. **变异体生成MutantProduction类**

该类用于根据BPEL程序的解析获得的可变异位置和变异算子列表，生成一阶或二阶变异体，方法如下：

* **GenerateFacade 类：**外观类，提供调用一阶和二阶变异体生成子系统接口，减少客户端和变异体生成子系统之间的耦合。
* **FirstGen类：**完成相应变异算子的一阶变异体生成，并记录变异体变异位置，使用的变异算子等信息。
* **SecondGen类：**完成相应变异算子的二阶变异体生成，并记录产生二阶变异体的基本信息。

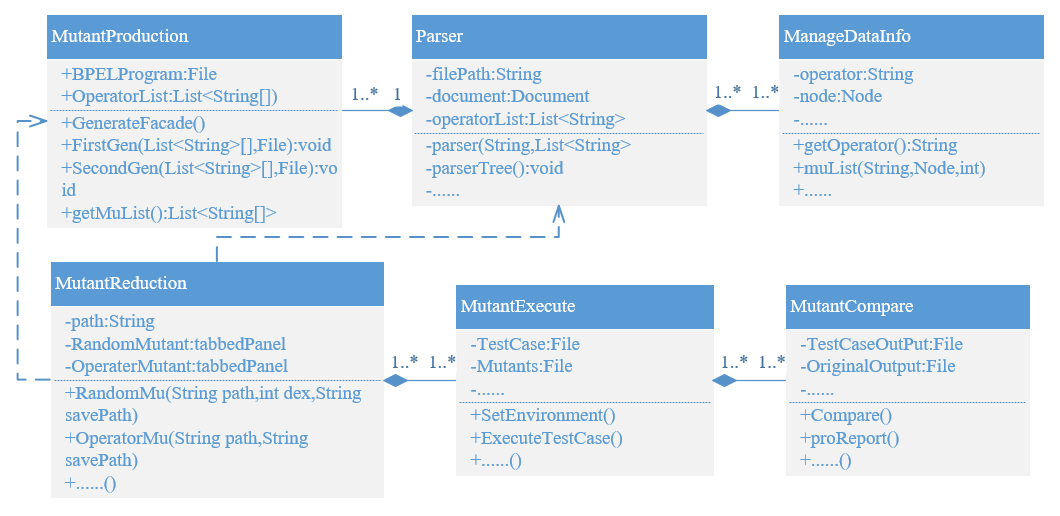


图4-5工具类图

1. **变异体精简MutationReduction类**

该类用于实现变异测试优化技术的业务逻辑，方法如下：

* **RandomMu()：**通过读入变异体程序所在路径信息及需要精简的比例参数，使用Random函数完成变异体的随机选择，得到变异体子集。最后，将选取的精简变异体子集存放在指定的文件夹中。
* **OperatorMu()：**该函数用于根据BPEL程序解析结果，获得的可以应用变异算子子集；然后，识别出具有包含关系的变异算子的剔除操作，获得精简的变异算子子集，保存在List中。调用变异体生成类，存入变异算子List，完成相应一阶变异体的生成。

1. **变异体执行MutantExecute类**

根据某优化技术选取的变异体子集设计测试用例，并将测试用例在所有变异体上执行，方法如下：

* **SetEnvironment():** 该方法用于配置BPEL执行环境，通过解析基于XML文件的配置信息，获取BPEL服务的端口号endpoint和操作名称operation name等操作。
* **ExecuteTestCase()：**该方法用于读取txt格式的测试用例文件，对第一行的用例信息，进行解析和识别，提取出输入参数个数和类别，并将用例保存在list数据结构中，依次向BPEL程序发送消息，自动执行变异体，将输出结果保存到以变异体名字加上“\_result”后缀命名的文件中。

1. **结果统计MutantCompare类**

对输入的原始程序和变异体测试结果进行统计分析，方法如下：

* **Compare():** 该方法通过读入原始程序和变异体测试结果文件，对变异体程序的每条测试用例结果提取和解析，与原始程序的输出结果进行对比，输出变异体被测杀情况。
* **proReport()：**该方法用于根据结果统计情况，输出此次变异测试的对象名称，变异体数量，变异得分等信息。
  1. 系统演示

μBPEL工具界面如图4-4所示，主要由四个部分组成：菜单栏、文件选择区域、具体功能区域和日志区域。其中，菜单部分提供工具的重启、Tomcat的运行、帮助三个菜单项；文件选择区域用来显示待测BPEL程序信息；具体功能区域是工具的主要操作点，包括变异体生成、变异体优化、测试用例生成、测试用例执行及结果分析五个部分；日志区域提供三种日志的显示：[GenerationLog]显示变异体生成时的日志；[ServerLog]显示Tomcat运行时输出日志；[LogInfo]显示与用户操作行为相关的日志。

采用SupplyCustomer[2]实例来演示μBPEL工具的使用，包括变异体生成，变异体优化、测试用例执行及结果分析部分。SupplyCustomer是一个项目订单管理的BPEL实例，涉及6个Web服务，其执行流程如图4-5所示。SupplyCustomer接收客户的两个输入参数，分别是订单的货物信息和地址，系统验证之后向客户反馈信息。

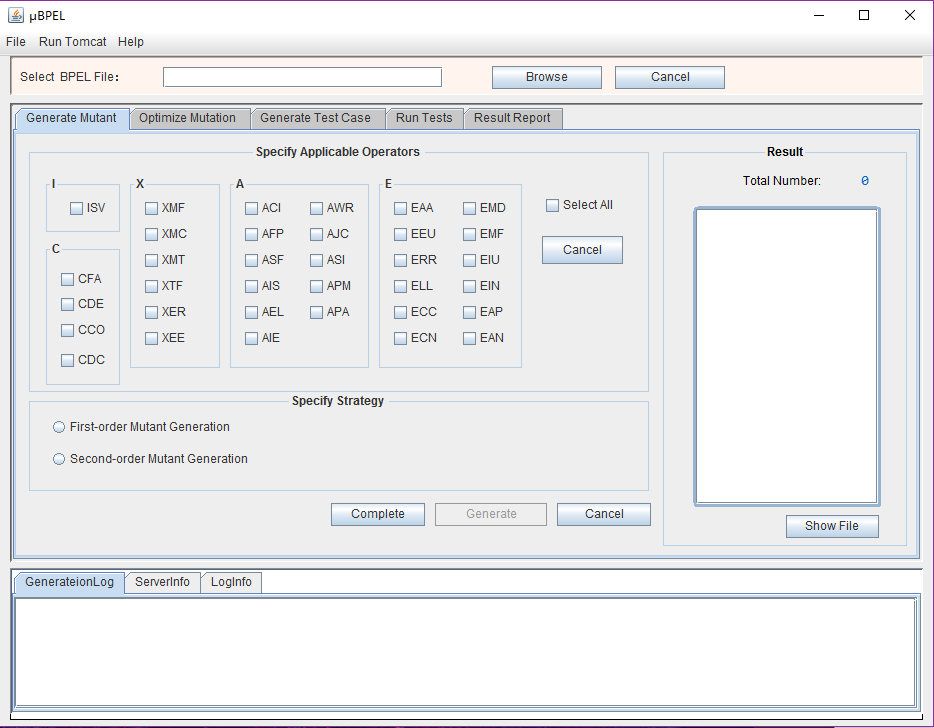


图4-6工具主界面

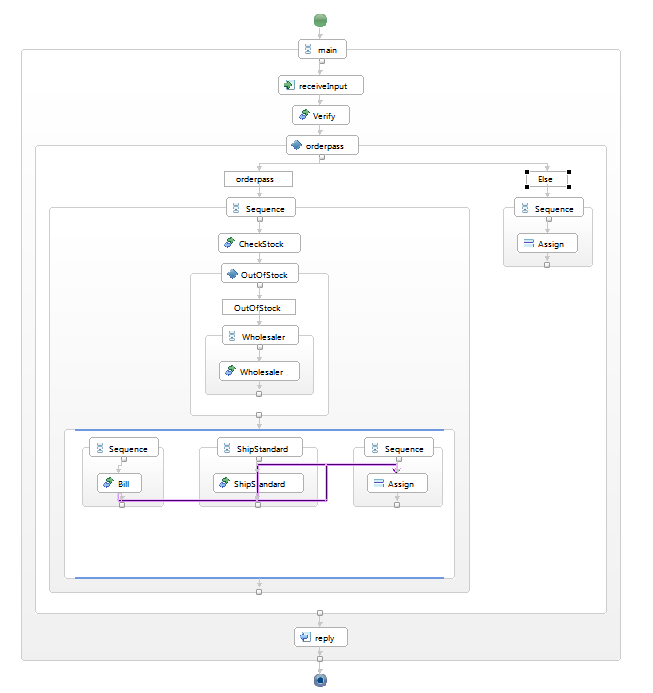


图4-7 SupplyCustomer实例执行流程

1. **变异体生成过程演示**

在BPEL程序的变异测试前，需要选入待测程序并开启Tomcat。开启Tomcat是用来支持BPEL程序的运行。用户可以在Configuration.xml配置文件中自定义Tomcat的文件路径。当Tomcat成功开启后，会在界面上显示“Open Tomcat Successfully”提示语。完成以上步骤，可以进行变异测试。 首先选择[Generate Mutant]标签，进入变异体生成界面。在[Specify Applicable Operators]框中选择需要应用的变异算子（或直接点击[Select All]选择所有变异算子）。其中，鼠标指向变异算子时，界面给出相应的变异算子的转换规则描述。通过[Specify Strategy]框，可以对变异体的种类进行选择，即生成一阶变异体还是二阶变异体。若选择[First-order Mutant Generation]按钮，并完成变异算子和策略的选择后，点击[Complete]按钮，系统会将所配置的选项加载进去。点击[Generate]按钮，系统会完成相应变异算子的一阶变异体生成。变异体生成结果显示在[Result]框中。如图4-6所示，SupplyCustomer共生成73个一阶变异体，变异体由所应用的变异算子名称加“\_”和数字命名。若在[Specify Strategy]框中，选择[Second-Order Mutant Generation]按钮，其他步骤同一阶变异体生成过程，系统将会生成二阶变异体。图4-7展示该实例的二阶变异体生成情况。

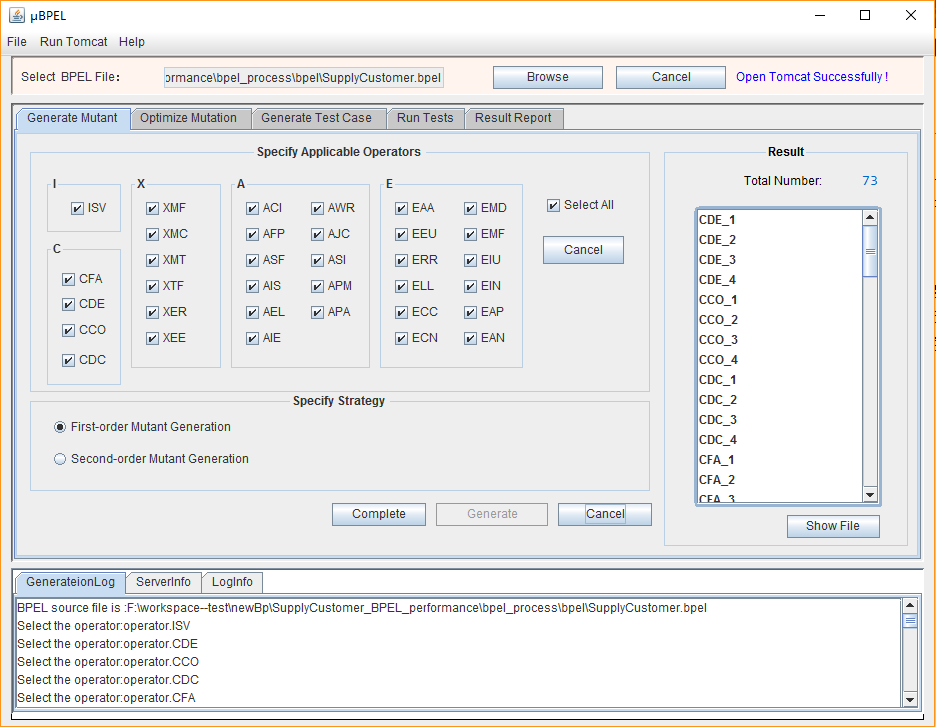


图4-8一阶变异体生成界面

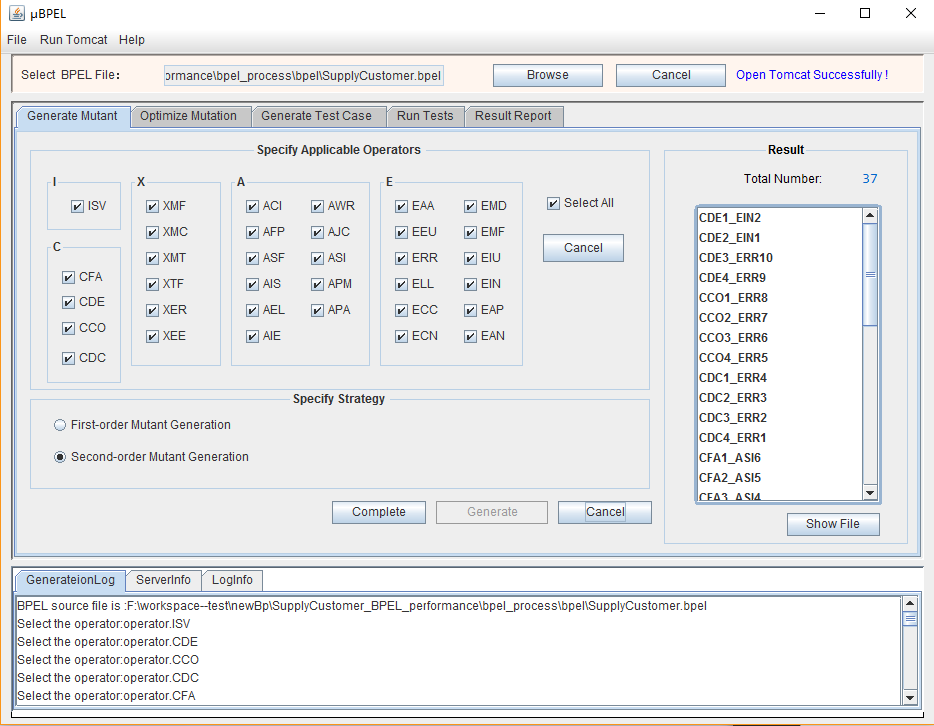


图4-9二阶变异体生成界面

为了方便查看变异体与原始程序的不同之处，系统提供一个[Mutants Display]界面对此功能进行支持。在界面中点击所要查看的变异体程序，界面底部的框中会给出该变异体变异的位置和内容，如图4-8所示。

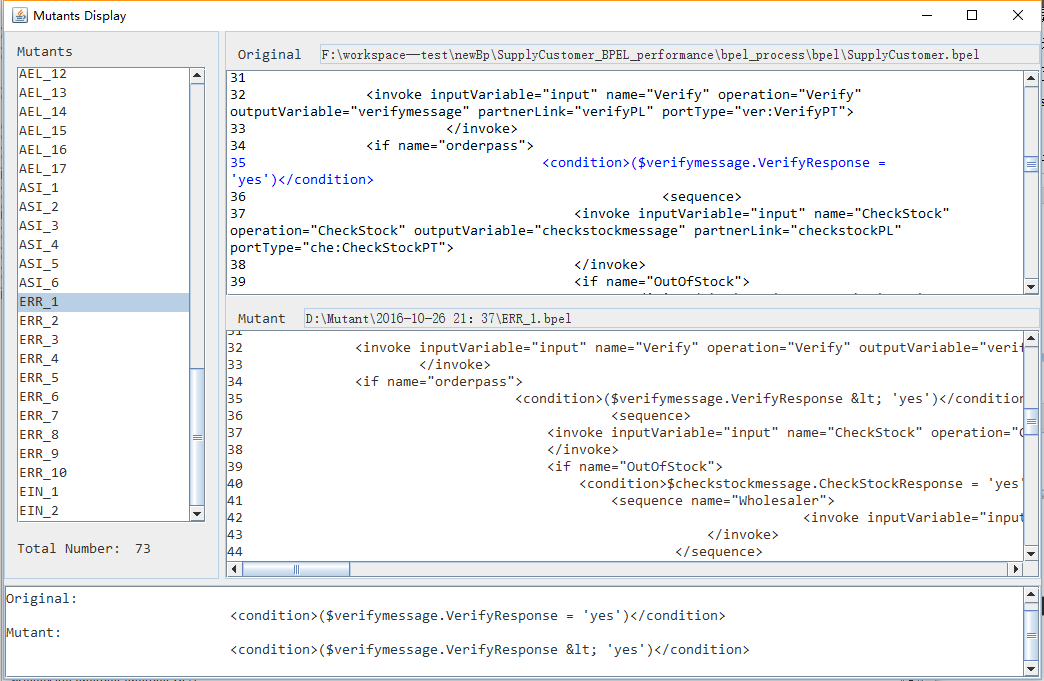


图4-10文件对比界面

1. **变异体优化演示**

通过点击[Optimize Mutation]菜单项，进入变异体优化界面。该模块提供两种变异体优化技术：一种是变异体随机选择优化技术，一种是基于包含关系的变异测试优化技术（MuSR）。具体如下：

1. **变异体随机选择优化技术演示**

点开[R-M]标签页，进入变异体随机选择优化界面。首先用户输入待优化的变异体所在文件夹的路径，系统会解析出该变异体集合的数目和所包含的变异体，结果[Result]中显示。然后，根据优化目标，输入约简比例，如图4-9所示。最后，点击[Generate]按钮，完成随机选择优化过程。本实例中，输入实例的一阶变异体所在文件夹路径和10%的选取比例；μBPEL工具从变异体集合中，随机的抽取10%的变异体，得到精简的变异体集合，相关信息显示在[Result]中，如图4-10所示。

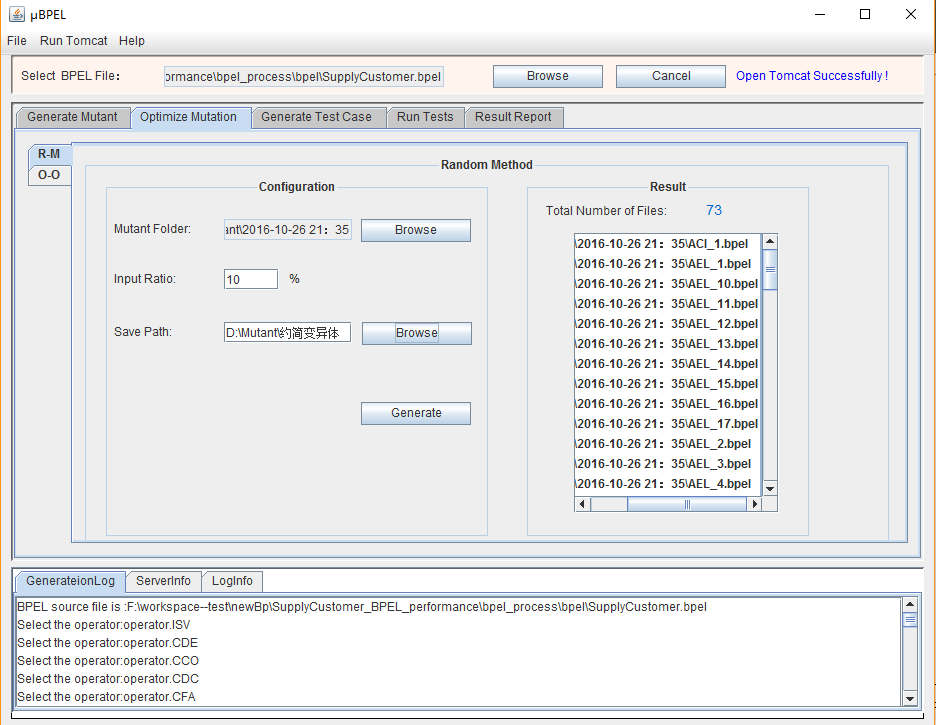


图4-11随机优化方法配置

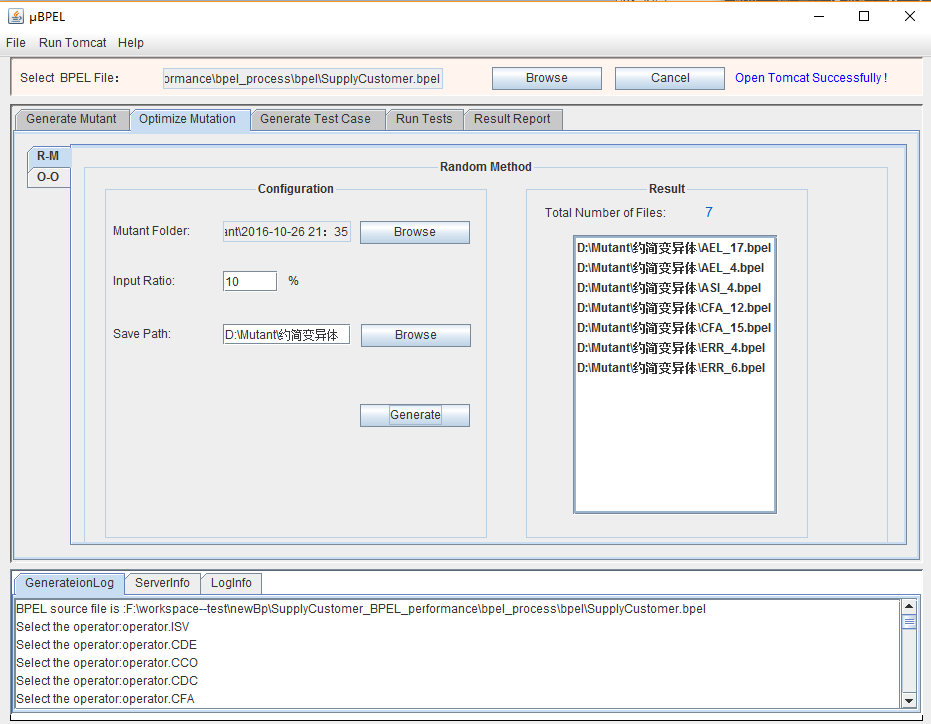
****

图4-12 随机抽取变异体执行过程

1. **基于包含关系的变异测试优化技术演示**

点开[O-O]标签页，进入界面。首先点击[Parse]按钮，μBPEL会自动识别出程序可以应用的变异算子。如图所示4-11所示，SupplyCustomer实例可以应用11种变异算子。点击[Optimize]按钮，基于MuSR技术，系统自动识别出这些被包含的变异算子，并将被包含的变异算子从变异算子集合中剔除（这里表现为取消选中）。在本实例中，*ASI*、*EIN*、*AIE*、*CFA*、*CCO*、*CDE*这6个变异算子被踢除。点击[Generate]按钮，系统会对剩下的5种变异算子生成变异体，得到的变异体信息在[Result]文本框中显示，如图4-12所示，该实例仅生成39个变异体。

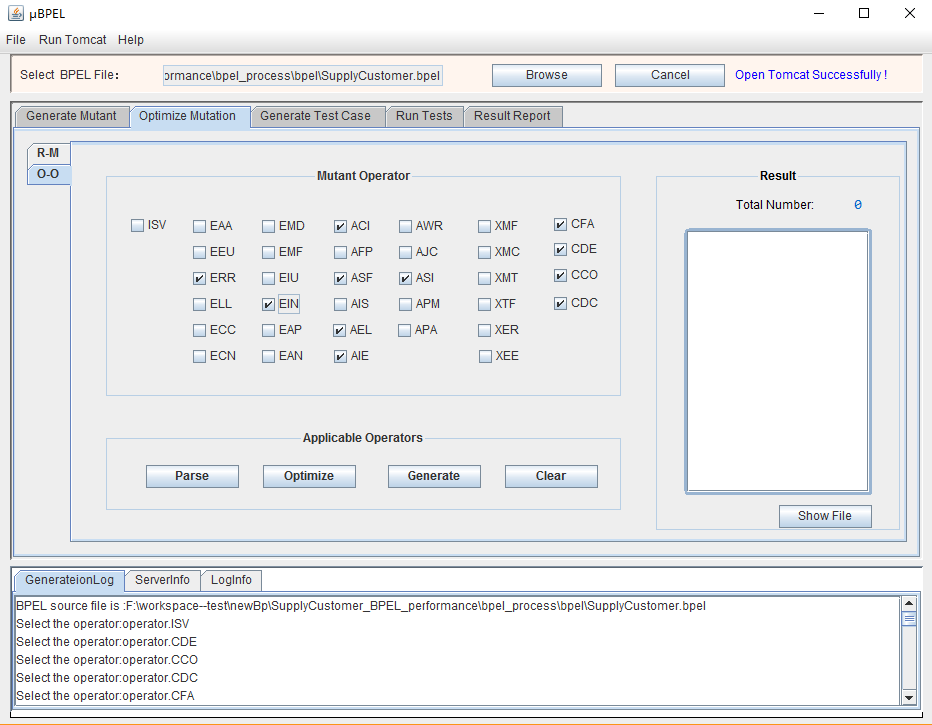


图4-13基于包含关系的变异体优化技术界面

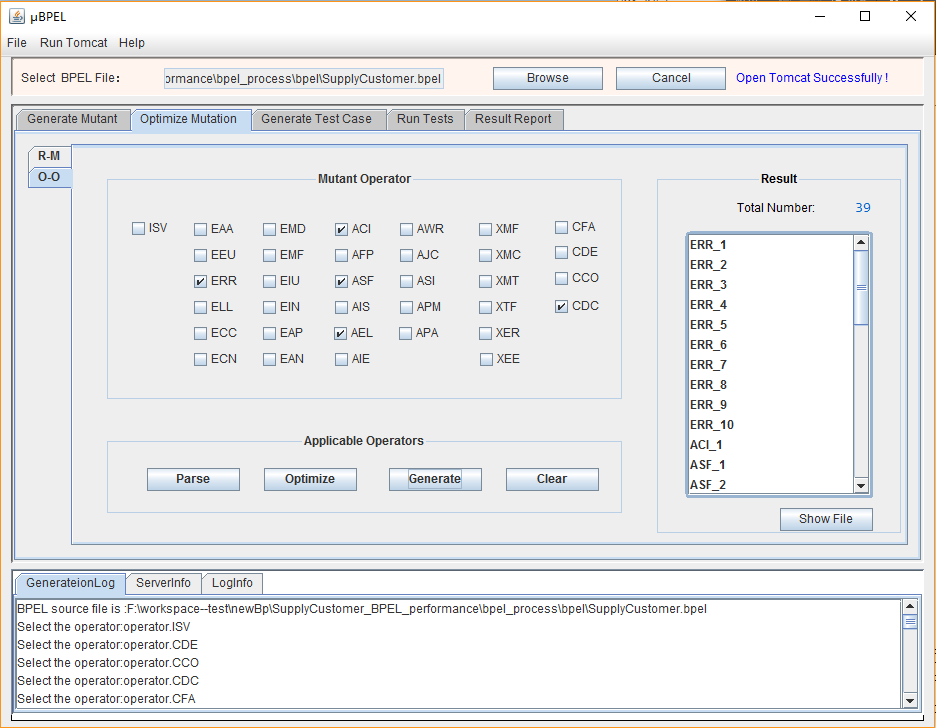


图4-14基于包含关系的变异体优化技术执行结果

1. **测试执行演示**

通过以上步骤，得到变异体集合和测试用例集合，可以进行变异测试。通过[Run Tests]标签页，进入执行测试界面。在对BPEL程序执行测试前，需要先将其部署到Tomcat上，μBPEL通过build.xml脚本来完成BPEL服务的部署。然后，输入待执行的BPEL程序和测试用例文件，完成基本测试环境的配置。点击[Run Test Cases]按钮，μBPEL将会自动获取程序的输入端口等信息，执行测试用例并将测试结果保存在文件中，如图4-13所示。

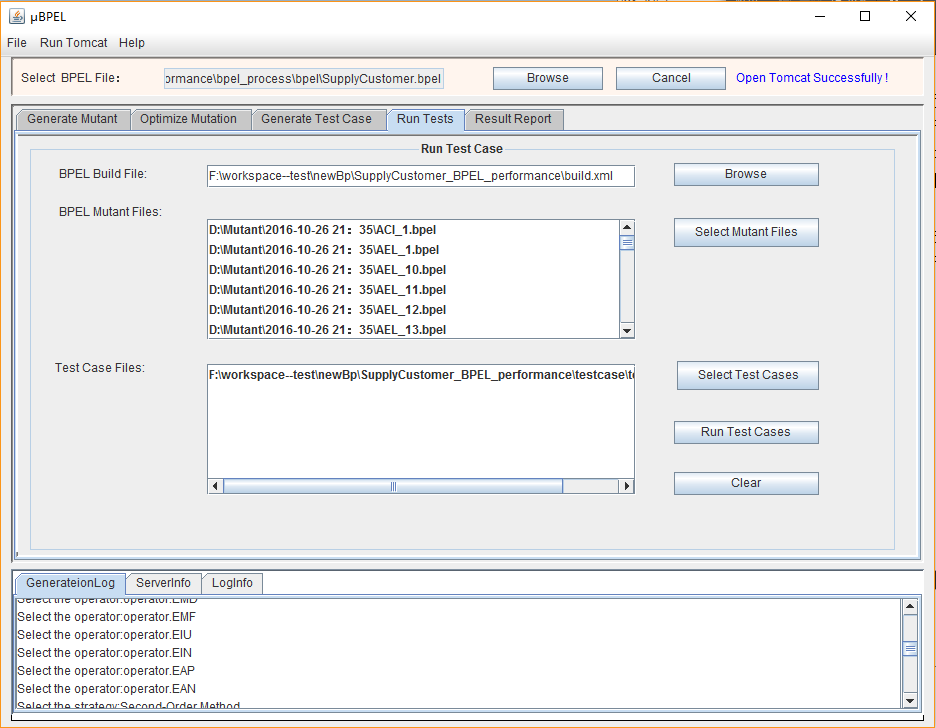


图4-15测试用例执行界面

1. **测试结果分析演示**

完成测试用例的执行，还需要对测试结果进行统计。通过点击[Result Report]菜单项，进入结果分析界面。点击[Browse]按钮选择原始程序测试结果和[Select Mutatnt Output]按钮选取所有变异体的测试用例执行结果，单击[Run]按钮对它们的测试结果进行对比，并输出结果。

系统统计变异体的故障检测率并显示在界面上。如图4-14所示，变异体的测试状态会在[Status]的项中显示，其中，“F”表示该条测试用例杀死变异体；“T”表示该条测试用例没有杀死变异体。[FinalStatus]状态栏显示该组测试用例集合是否杀死”变异体，其中“SUCCESS”表示变异体存活；“KILLED”表示变异体被“杀死”。[Result]栏中显示实例的变异测试报告。本实例中，被杀死的变异体数量共59个、总变异体数量共73个，最终的变异得分为81%。

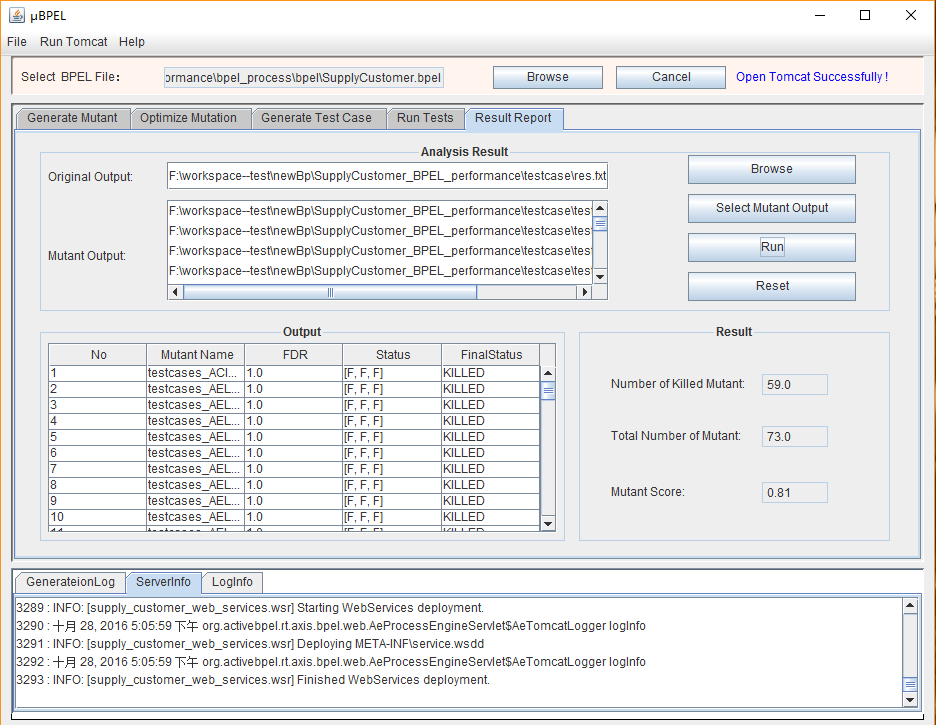


图4-16 测试执行结果

* 1. 小结

本章详细讨论了BPEL程序的变异测试集成化工具μBPEL的设计与实现。μBPEL 支持BPEL程序变异测试的全过程，同时实现了本文提出的部分变异测试优化技术，有助于测试人员对BPEL进行高效的变异测试。

1. 经验研究

本章对课题组前期的经验研究进行扩充，以增强实验结果的可靠性；采用经验研究评估所提出的优化技术的有效性。

自己的：实验中使用SOAP消息对测试用例进行封装，并通过HTTP进行数据传输，将SOAP请求发送到该服务所在的服务器，处理流程包括的测试用例的生成选择，测试执行和测试结果分析。

* 1. 实验对象与度量指标

1. **实验对象**

本文实验对象共包括6个BPEL程序实例。SupplyChain[10]是一个供销链管理的实例。客户输入商品的名字和数量，零售商将会根据库存的状况反馈信息。SmartShelf[10]是一个商品货架管理的实例。用户输入货物的相关信息，例如货物名称、数量，系统处理这些信息并输出商品库存数量、货架位置和库存状态三个信息。SupplyCustomer[2]是项目订单管理的实例。用户输入订单的名称和地址，系统处理并反馈订单的查询结果。LoanApproval[2]是贷款审批的实例。用户输入个人信息及贷款金额，系统进行处理并反馈贷款审核结果。CarEstimate[41]是用于汽车评估的实例。用户发出评估请求，系统提供初步、简单、复杂评估方式，并返回最终评估结果。TravelAgency[42]是一个旅行社预订实例，组合旅店预订、订票、旅行社预订、和银行结算服务。用户提供预订旅客的信息和人数，系统进行处理并反馈结果。表5-1给出这些实例的具体信息，包括实例的基本功能描述（Basic functionality）、组装的服务数目（No. of services composed）和实例的规模（Size）。文中，使用P1至P6依次代表SupplyChain、SmartShelf、SupplyCustomer、LoanApproval、CarEstimate和TravelAgency实例程序。

表5-1 BPEL的实例程序基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| **Program** | **Basic functionality** | **No. of services composed** | **Size (LOC)** |
| P1 | Management of supply chains | 2 | 50 |
| P2 | Management of commodity shelves | 14 | 194 |
| P3 | Management of project orders | 5 | 122 |
| P4 | Examination of loan applications | 3 | 120 |
| P5 | Assessment of car repairs | 7 | 121 |
| P6 | Booking of travels | 9 | 543 |

1. **度量指标**

使用如下三个指标来度量变异测试优化方法的有效性。

1. **变异得分**（Mutation Score）

定义测试用例集杀死的非等价变体的比例[4]，用来衡量测试的充分程度。变异得分计算公式如下：

 (5-1)

其中，*P*代表被测程序，*TS*代表测试数据集，*Nk*表示被杀死的变异体数量，*Nm*表示变异体总数量，*Ne*代表等价变异体的数量。变异得分越高，说明越多的非等价变异体被测试用例集“杀死”，测试用例集越有效。

1. **故障检测率**

定义测试用例能够有多少比例杀死变异体，具体信息在文章3.3进行了介绍。

1. **APFD**（Average of the Percentage of Faults Detected）

一种基于错误的评价标准，多用于根据测试集的错误检测效率来衡量不同优先级技术的排序效果[43]。APFD的计算公式如下：

 (5-2)

其中，*TS*表示具有顺序的测试用例集，*P*表示待测程序，*n*表示测试用例的个数，*m*表示被检测错误的总数，*reveal(i, TS)*表示最早检测出错误*i*所执行的测试用例的位置。APFD量化了测试用例序列的效率和效能[44]。本文采用APFD来度量序列化的测试用例的故障检测效率。APFD的数值越大，说明该测试用例集检测错误速率越快、效率越高。

* 1. 扩充的实例研究

对课题组面向BPEL的变异测试的实验进行扩充，以增强相关研究成果的可靠性。

* + 1. 实验步骤

对实验对象进行变异测试，实验步骤如下：

1. 选取一个实例程序*P*，使用μBPEL为该程序生成一阶变异体；将这些一阶变异体组成集合，记为*M*。
2. 使用不同用例生成技术构造测试用例集合*TS*。在实例程序*P*和一阶变异体集合*M*上运行测试用例集合*TS*，记录测试结果。
3. 分析测试结果，找出没有被测试用例集*TS*“杀死”的一阶变异体，识别出其中的等价变异体。
4. 使用μBPEL 统计故障检测率FDR和变异得分MS。
   * 1. 实验结果
5. **变异体生成**

使用μBPEL为实例自动化生成一阶变异体，结果如表5-2所示。

表5-2 BPEL实例的变异体生成

|  |  |  |
| --- | --- | --- |
| **Program** | **No. of generated first-order mutants** | **No. of used operators** |
| P1 | 34 | 11 |
| P2 | 170 | 11 |
| P3 | 73 | 11 |
| P4 | 85 | 17 |
| P5 | 54 | 5 |
| P6 | 171 | 14 |

1. **测试用例生成**

课题组前期工作中主要使用了等价类划分[45]和边界值分析[45]两种技术生成测试用例。为了减少不同测试用例生成技术对变异算子的经验研究成果的影响，本文新增两种测试用例生成技术：面向场景测试技术[36]和随机测试技术[45]，来增加实验结果的通用性。面向场景的测试技术首先将BPEL程序转换为图模型结构，获取测试场景，最后根据不同场景产生用例。随机测试以输入和约束的范围生成用例。前期工作产生的三组测试用例集合分别命名为Tx、Ty和Tz，新增的技术产生的测试用例集，命名为Tu和Tw 。表5-3给出每个BPEL实例的测试用例集数量。需要指出的是，测试用例集的数量依赖于程序的规模和所使用的测试用例生成技术。对于小规模的程序所产生的用例集的数量相对较少。例如，SupplyChain和SmartShelf实例分别由最少和最多数量的服务组装而成，它们所关联的用例数量也是最少和最多的；另一方面，不同的测试技术也会导致不同测试用例的产生。

表5-3 BPEL实例的测试用例集合

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Program** | **|Tx|** | **|Ty|** | **|Tz|** | **|Tu|** | **|Tw|** |
| P1 | 6 | 10 | 15 | 19 | 30 |
| P2 | 22 | 35 | 50 | 74 | 90 |
| P3 | 7 | 12 | 18 | 24 | 30 |
| P4 | 12 | 18 | 25 | 31 | 40 |
| P5 | 10 | 20 | 30 | 36 | 40 |
| P6 | 4 | 7 | 10 | 14 | 20 |

1. **测试结果**

对实验结果进行统计，计算故障检测率(FDR)和变异得分(MS)，结果如表5-4和5-5所示。表5-4给出不同测试用例集合下，变异体的FDR值范围，其中NA表示由此变异算子产生的变异体都是等价变异体，无法统计FDR值。表5-5给出实例变异测试的变异得分。

表5-4 6个BPEL实例的故障检测率FDR

**(a) SupplyChain**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ACI | {100%,100%} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| AEL | {50%,100%} | {40%,100%} | {40%,100%} | {42.1%100%} | {36.7%,100%} |
| AIE | {50%,50%} | {60%,60%} | {60%,60%} | {57.9%,57.9} | {63.3%,63.3} |
| ASI | {50%,100%} | {40%,100%} | {40%,100%} | {42.1%,100%} | {36.7%,100%} |
| ASF | {50%,100%} | {40%,100%} | {40%,100%} | {42.1%,100%} | {36.7%,100%} |
| ERR | {16.7%,100} | {10%,100%} | {6.7%,100%} | {5.3%,100%} | {3.3%,100%} |
| EIN | {100%,100%} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| CFA | {50%,100%} | {40%,100%} | {40%,100%} | {42.1%100%} | {36.7%,100%} |
| CDE | {50%,50%} | {40%,60%} | {40%,60%} | {42.1%,57.9} | {36.7%,63.3} |
| CDC | {50%,50%} | {40%,60%} | {40%,60%} | {42.1%,57.9} | {36.7%,63.3} |
| CCO | {50%,50%} | {40%,60%} | {40%,60%} | {42.1%,57.9} | {36.7%,63.3} |

**(b) SmartShelf**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ACI | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} |
| AEL | { 18.2%,100%} | {20%,100%} | {20%,100%} | {21.6%,100%} | {24.4%,100%} |
| AIE | {40.9%,81.9%} | {37.1%,80%} | {36%,80%} | {33.8%,82.4%} | {30%,75.6%} |
| ASI | { 18.2%,100%} | {20%,100%} | {20%,100%} | {21.6%,100%} | {24.4%,100%} |
| ASF | { 18.2%,100%} | {20%,100%} | {20%,100%} | {21.6%,100%} | {24.4%,100%} |
| ERR | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} |
| EIN | {59.1%,100%} | {60%,100%} | {64%,100%} | {63.5%,100%} | {66.7%,100%} |
| CFA | { 18.2%,100%} | {20%,100%} | {20%,100%} | {21.6%,100%} | {24.4%,100%} |
| CDE | {18.2%,81.9%} | {20%,80%} | {20%,80%} | {21.6%,78.4%} | {24.4%,75.6%} |
| CCO | {18.2%,81.9%} | {20%,80%} | {20%,80%} | {21.6%,78.4%} | {24.4%,75.6%} |
| CDC | {18.2%,81.9%} | {20%,80%} | {20%,80%} | {21.6%,78.4%} | {24.4%,75.6%} |

**(c) SupplyCustomer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ACI | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} |
| ASF | {57.1%,100%} | {50%,100%} | {50%,100%} | {50%,100%} | {46.7%,100%} |
| AIE | {42.9%,42.9} | {50%,50%} | {50%,50%} | {50%,50%} | {53.3%,53.3} |
| ASI | {14.3%,100%} | {8.3%,100%} | {11.1%,100%} | {12.5%,100%} | {13.3%,100%} |
| AEL | {14.3%,100%} | {8.3%,100%} | {11.1%,100%} | {12.5%,100%} | {13.3%,100%} |
| ERR | {14.3%,100%} | {8.3%,100%} | {11.1%,100%} | {12.5%,100%} | {13.3%,100%} |
| EIN | {57.1%,100%} | {50%,100%} | {50%,100%} | {50%,100%} | {46.7%,100%} |
| CFA | {14.3%,100%} | {8.3%,100%} | {11.1%,100%} | {12.5%,100%} | {13.3%,100%} |
| CDE | {14.3%,85.7} | {8.3%,91.7%} | {11.1%,88.9} | {12.5%,87.5%} | {13.3%,86.7} |
| CCO | {14.3%,85.7} | {8.3%,91.7%} | {11.1%,88.9} | {12.5%,87.5%} | {13.3%,86.7} |
| CDC | {14.3%,85.7} | {8.3%,91.7%} | {11.1%,88.9} | {12.5%,87.5%} | {13.3%,86.7} |

**(d) CarEstimate**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ACI | {100%,100} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| AEL | {100%,100} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| ASI | {100%,100} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| ASF | {100%,100} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |
| CFA | {100%,100} | {100%,100} | {100%,100} | {100%,100%} | {100%,100%} |

**(e) LoanApproval**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ISV | {8.3%,8.3%} | {11.1%,11.1} | {12%,12%} | {12.9%,12.9%} | {12.5%,12.5} |
| ACI | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} |
| AEL | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} |
| ASI | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} | {100%,100%} |
| ASF | {8.3%,8.3%} | {11.1%,11.1} | {12%,12%} | {12.9%,12.9%} | {12.5%,12.5} |
| ECN | {8.3%,50%} | {5.6%,50%} | {4%,48%} | {3.2%,38.7%} | {2.5%,32.5%} |
| ERR | {8.3%,83.3%} | {5.6%,77.8%} | {4%,72%} | {3.2%,71%} | {2.5%,65%} |
| EIU | {25%,75%} | {22.2%,66.7} | {20%,50%} | {19.4%,58.1%} | {17.5%,52.5} |
| EIN | {8.3%,83.3%} | {11.1%,77.8} | {12%,72%} | {12.9%,71%} | {12.5%,65%} |
| EAN | {25%,75%} | {22.2%,66.7} | {20%,50%} | {19.4%,58.1%} | {17.5%,52.5} |
| ECC | NA | NA | NA | NA | NA |
| EAP | NA | NA | NA | NA | NA |
| XMF | {0%,0%} | {0%,0%} | {0%,0%} | {0%,0%} | {0%,0%} |
| CFA | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} | {0%,100%} |
| CDE | {8.3%,75%} | {11.1%,66.7} | {12%,60%) | {12.9%,58.1%} | {12.5%,52.5} |
| CCO | {8.3%,75%} | {11.1%,66.7} | {12%,60%) | {12.9%,58.1%} | {12.5%,52.5} |
| CDC | {8.3%,75%} | {11.1%,66.7} | {12%,60%) | {12.9%,58.1%} | {12.5%,52.5} |

**(f) TravelAgency**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Oper**  **-ator** | **Value range of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ACI | {100%,100} | {100%,100%} | {100%,100} | {100%,100} | {100%,100} |
| AEL | {50%,100%} | {42.9%,100%} | {50%,100%} | {50%,100%} | {45%,100%} |
| AIE | {50%,50%} | {57.1%,57.1} | {50%,50%} | {50%,50%} | {45%,45%} |
| ASI | {50%,100%} | {42.9%,100%} | {50%,100%} | {50%,100%} | {45%,100%} |
| ASF | {50%,100%} | {42.9%,100%} | {50%,100%} | {50%,100%} | {45%,100%} |
| ERR | {25%,100%} | {14.3%,,100} | {%10,100%} | {7.1%,100%} | {5%,100%} |
| ECN | {25%,50%} | {14.3%,57.1} | {10%,50%} | {7.1%,50%} | {5%,50%} |
| EAN | {50%,50%} | {42.9%,42.9} | {50%,50%} | {50%,50%} | {55%,55%} |
| EIU | {50%,50%} | {42.9%,42.9} | {50%,50%} | {50%,50%} | {55%,55%} |
| EIN | {100%,100} | {100%,100%} | {100%,100} | {100%,100} | {100%,100} |
| ECC | NA | NA | NA | NA | NA |
| EAP | NA | NA | NA | NA | NA |
| CFA | {50%,100%} | {42.9%,100%} | {50%,100%} | {50%,100%} | {45%,100%} |
| CDE | {50%,50%} | {42.9%,57.1} | {50%,50%} | {50%,50%} | {45%,55%} |
| CCO | {50%,50%} | {42.9%,57.1} | {50%,50%} | {50%,50%} | {45%,55%} |
| CDC | {50%,50%} | {42.9%,57.1} | {50%,50%} | {50%,50%} | {45%,55%} |

表5-5 6个BPEL实例的变异得分MS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Program** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** |
| **MS** | 100% | 99.3% | 100% | 96.3% | 100% | 100% |

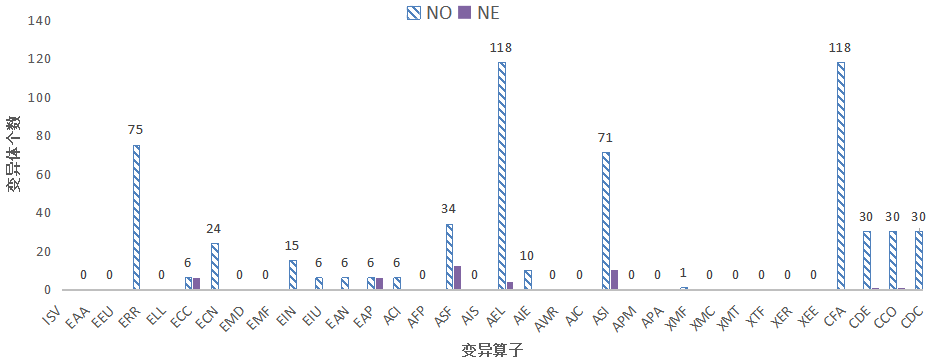


图5-1变异算子产生变异体情况

基于以上实验结果，统计不同变异算子产生变异体的情况，如图5-1所示。在图5-1中，*NO*指由此变异算子产生的变异体数目，*NE*指由此变异算子产生的等价变异体数目。通过分析这些数据，可以看出：

1. EAA，EEU，ELL，EMD，EMF，AFP，AIS，AWR，AJC，APM, APA, XMC，XMT，XTF，XER，XEE这16种变异算子在这6个BPEL实例中不适用。
2. AEL，CFA，ERR，ASI这四类变异算子相对于其他变异算子可产生较多变异体。
3. ECC，EAP这两种变异算子产生的变异体都是等价变异体；ASF， ASI，AEL这三类变异算子会产生大量的等价变异体。
   1. 变异测试优化技术实例评估

采用经验研究方法验证和评估本文提出的优化技术的有效性。

* + 1. MuSR技术评估

在文章3.2节，提出了变异算子的包含关系规则，并总结出7组具有包含关系的变异算子。本部分实验运用包含关系规则验证这7组包含关系的正确性。同时，评估MuSR技术对于BPEL程序的变异测试优化的效果。

根据包含关系规则定义可知，若*OA←OB*，则说明对于变异算子*OB*产生的所有变异体，都存在变异算子*OA*产生的变异体，使得所有能将*OA*产生的变异体“杀死”的测试用例，都能“杀死”*OB*产生的变异体。*OB*产生的变异体可以被*OA*的变异体所替代，所以*OB*的变异体可以被精简。

通过查询5.2节中实例的变异测试结果，统计7组具有包含关系的变异算子产生的变异体的FDR值，观察它们的测试情况。FDR值给出测试用例能够有多少比例“杀死”变异体，可以间接反映出变异算子间的包含关系。若*AEL←AIE*，根据包含关系定义，可知*AEL*变异体的FDR值应小于等于*AIE*变异体的FDR值。对每组变异算子之间的包含关系找一个BPEL实例进行验证，得到如表5-6至5-11所示结果。表5-6至5-11给出在五组测试用例集合下，这些具有包含关系的变异算子产生变异体的FDR值。其中，表5-7展示SupplyCustomer实例中存在*AEL←AIE*这组包含关系。在五组测试用例集合下，*AEL*和*AIE*的变异体的FDR值相同，表明所有能将*AEL*的*MU1AEL*或*MU2AE*L变异体的“杀死”的测试用例，都能将*AIE*变异算子的*MU1AIE*变异体“杀死”。所以，*AIE*和*AEL*满足包含规则，*AEL←AIE*成立。同理，其余6组变异算子的包含关系也成立。

表5-6 SupplyChain的ASI和ASF算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| ASI | MU1ASI | 100% | 100% | 100% | 100% | 100% |
| MU2ASI | 100% | 100% | 100% | 100% | 100% |
| MU3ASI | 100% | 100% | 100% | 100% | 100% |
| MU4ASI | 50% | 40% | 40% | 42.1% | 36% |
| ASF | MU1AEL | 100% | 100% | 100% | 100% | 100% |
| MU2AEL | 50% | 40% | 40% | 42.1% | 36% |

表5-7 SupplyCustomer的AIE和AEL算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| AIE | MU1AIE | 42.9% | 50% | 50% | 50% | 53.3% |
| AEL | MU1AEL | 42.9% | 50% | 50% | 50% | 53.3% |
| MU2AEL | 42.9% | 50% | 50% | 50% | 53.3% |

表5-8 SupplyChain的CCO、CDE和CDC算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| CCO | MU1CCO | 50% | 40% | 40% | 42.1% | 36.7% |
| MU2CCO | 50% | 60% | 60% | 57.9% | 63.3% |
| CDE | MU1CDE | 50% | 40% | 40% | 42.1% | 36.7% |
| MU2CDE | 50% | 60% | 60% | 57.9% | 63.3% |
| CDC | MU1CDC | 50% | 40% | 40% | 42.1% | 36.7% |
| MU2CDC | 50% | 60% | 60% | 57.9% | 63.3% |

表5-9 SupplyChain的CFA和AEL算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| CFA | MU1CFA | 100% | 100% | 100% | 100% | 100% |
| MU2CFA | 100% | 100% | 100% | 100% | 100% |
| MU3CFA | 50% | 40% | 40% | 42.1% | 36.7% |
| MU4CFA | 50% | 40% | 40% | 42.1% | 36.7% |
| MU5CFA | 50% | 40% | 40% | 42.1% | 36.7% |
| MU6CFA | 50% | 60% | 60% | 57.9% | 63.3% |
| MU7CFA | 100% | 100% | 100% | 100% | 100% |
| AEL | MU1AEL | 100% | 100% | 100% | 100% | 100% |
| MU2AEL | 100% | 100% | 100% | 100% | 100% |
| MU3AEL | 50% | 40% | 40% | 42.1% | 36.7% |
| MU4AEL | 50% | 40% | 40% | 42.1% | 36.7% |
| MU5AEL | 50% | 40% | 40% | 42.1% | 36.7% |
| MU6AEL | 50% | 60% | 60% | 57.9% | 63.3% |
| MU7AEL | 100% | 100% | 100% | 100% | 100% |

表5-10 TravelAgency的EIN和ERR算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| EIN | MU1EIN | 100% | 100% | 100% | 100% | 100% |
| MU2EIN | 100% | 100% | 100% | 100% | 100% |
| MU3EIN | 100% | 100% | 100% | 100% | 100% |
| MU4EIN | 100% | 100% | 100% | 100% | 100% |
| ERR | MU1ERR | 100% | 100% | 100% | 100% | 100% |
| MU2ERR | 100% | 100% | 100% | 100% | 100% |
| MU3ERR | 100% | 100% | 100% | 100% | 100% |
| MU4ERR | 100% | 100% | 100% | 100% | 100% |

表5-11 LoanApproval的EAN和EIU算子情况

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Mutant** | **Value of FDR** | | | | |
| Tx | Ty | Tz | Tu | Tw |
| EAN | MU1EAN | 75% | 66.7% | 50% | 58.1% | 52.5% |
| MU2EAN | 25% | 22.2% | 20% | 19.4% | 17.5% |
| EIU | MU1EIU | 75% | 66.7% | 50% | 58.1% | 52.5% |
| MU2EIU | 25% | 22.2% | 20% | 19.4% | 17.5% |

统计6个BPEL实例中与这7组包含关系关联的变异体，依据MuSR技术可知，这些变异体可以被精简。结果如表5-12所示。其中，属于*AEL←AIE*的变异体有11个，属于*ASF ← ASI*的变异体有61个，属于*AEL←CFA*的变异体有118个，属于*ERR ← EIN*的变异体有15个，属于*EIU ← EAN*的变异体有6个，属于*CDC ← CDE和CDC ←CCO*的变异体各有30个，共有271个变异体可被精简。由包含规则的定义可知，这271个变异体所代表的错误完全可以被其它变异体所替代。精简这部分变异体，可以减少46.2%比例的变异体开销。由此可见，MuSR技术极大的减少了待测变异体的数目，提高了测试效率。

表5-12 6个BPEL程序中具有包含关系变异算子组合的总结

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Program** | | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** | **Total** |
| **No. of**  **mutants pairs** | AEL←AIE | 1 | 4 | 1 | 0 | 0 | 5 | 11 |
| ASF←ASI | 4 | 19 | 6 | 1 | 15 | 16 | 61 |
| AEL←CFA | 7 | 40 | 17 | 8 | 16 | 30 | 118 |
| ERR←EIN | 1 | 4 | 2 | 4 | 0 | 4 | 15 |
| EIU←EAN | 0 | 0 | 0 | 2 | 0 | 4 | 6 |
| CDC←CCO | 2 | 8 | 4 | 8 | 0 | 8 | 30 |
| CDC←CDE | 2 | 8 | 4 | 8 | 0 | 8 | 30 |
| **Total** | | 17 | 83 | 34 | 31 | 31 | 75 | 271 |
| **Reduction** | | 50% | 48.8% | 46.6% | 36.5% | 57.4% | 43.8% | 46.2% |

* + 1. MuSOM技术评估

1. **实验步骤**

对BPEL实例的一阶变异体和二阶变异体进行变异测试，来评估MuSOM技术的有效性。图5-2给出具体的实验流程。



图5-2 二阶变异体测试过程

实验过程描述如下：

1. 输入实例源程序*P*，使用μBPEL工具来自动化生成变异体，得到一阶变异体集合，记为*M1o*和二阶变异体集合，记为*M2o*；
2. 选取5.2节中的测试用例集*Tx*，*Tx={T0,T1, …,Ti,…,Tn-1}*；令*n*=*|Tx|*，*i=0*；
3. 用*Ti*执行所有的二阶变异体和原始程序*P*，将结果保存在文件中；比较二阶变异体和原始程序的输出结果，若二者不同，说明*Ti*“杀死”该二阶变异体；否则，说明该变异体“存活”；
4. 若*M2o*中存在被*Ti*“杀死”的二阶变异体，则将*Ti*加入到测试用例集合*TC*中，并从*M2o*中删除这些被*Ti*“杀死”的二阶变异体；判断*M2o*集合是否为空，如果不是，则执行第五步；否则，则执行第八步；
5. 判断*i*是否等于*n*，如果不是，则令*i＝i+1*，返回第三步；否则，执行第六步；
6. 人工识别*M2o*中的等价变异体，将这些等价变异体从*M2o*中删除并加入到集合*E*中，执行下一步；
7. 判断*M2o*集合是否为空，如果不是，则手动添加能将*M2o*中变异体“杀死”的测试用例，并将这些测试用例添加到*TC*；否则，执行第八步；
8. 用测试用例集合*TC*，执行一阶变异体*M1o*，保存测试结果；
9. 使用μBPEL工具统计一阶变异体的变异得分*MS*。
10. **实验结果**

按照上小节的实验步骤，对实例进行实验。实验结果如表5-13和图5-3所示。表5-13给出实验中产生的一阶变异体和二阶变异体情况。其中，*P*指待测的原始程序；*Mutants*列表示使用μBPEL工具生成的一阶变异体集合*M1o*和二阶变异体集合*M2o*；*NS*表示应用的变异算子数目；*NM*表示产生的变异体程序数量；*NE*表示等价变异体程序的数量。图5-3给出实例中的一阶变异体集合和二阶变异体集合的变异得分情况。

表5-13 6个BPEL程序的实验结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **P** | **Mutants** | **NS** | **NM** | **NE** |
| P1 | M1o | 11 | 34 | 0 |
| M2o | 11 | 17 | 0 |
| P2 | M1o | 11 | 170 | 13 |
| M2o | 11 | 85 | 0 |
| P3 | M1o | 11 | 73 | 5 |
| M2o | 11 | 37 | 0 |
| P4 | M1o | 17 | 85 | 7 |
| M2o | 17 | 43 | 0 |
| P5 | M1o | 5 | 54 | 7 |
| M2o | 5 | 27 | 0 |
| P6 | M1o | 14 | 171 | 16 |
| M2o | 14 | 86 | 0 |

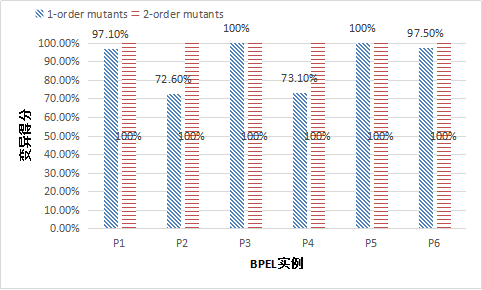


图5-3 6个BPEL程序的一阶与二阶变异体实验变异得分

通过实验结果，可以得出如下结论：

1. 在实验中，应用同样数目和种类的变异算子，*M2o*集合数量约为*M1o*的一半，减少了约50%的待测变异体。
2. *M2o*中的等价变异体数目（*NE*）总和为0，而*M1o*中存在48个等价变异体。可见，MuSOM技术大幅度降低等价变异体的识别开销。
3. 对于测试用例集*TC*，二阶变异体的变异得分均为100%，而一阶变异体的变异得分不尽相同。其中，*P3* 和*P5*实例的变异得分为100%；*P1*和*P6*实例的变异得分分别是97.1%和97.5%，接近于100%；而*P2*和*P4*实例的变异得分分别是72.6%和73.1%。可见，绝大多数一阶变异体可以被*TC*“杀死”。

综上所述，使用二阶变异体进行测试，相对于一阶变异体，可以减少近50%的变异体测试开销和8%的等价变异体的识别开销。同时，二阶变异体并没有大幅度降低衡量测试用例集故障检测的能力。

* + 1. MuPri技术评估

根据5.2节的实例的变异测试结果，计算变异算子的质量*Qo*，为变异算子进行优先级排序，得到表5-14结果。其中，“-”表示实例中该变异算子不适用；*Average*是对6个实例中的变异算子*Qo*值取得的平均值。根据这个值的大小为变异算子排序并分配优先级，优先级显示在*Rank*列中。Rank中的数值越小，表示该变异算子的优先级越高。优先级高的变异算子表示其产生的变异体越难被杀死，在检验测试用例有效性方面效果更好。需要说明的是，由于在这6个实例程序中，只有16个变异算子可适用，所以，仅对这16个变异算子进行了排序。

表5-14 变异算子排序

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Operator** | **Qo** | | | | | | **Average** | **Rank** |
| P1 | P2 | P3 | P4 | P5 | P6 |
| XMF | - | - | - | 100.0% | - | - | 100.0% | 1 |
| ISV | - | - | - | 88.6% | - | - | 88.6% | 2 |
| ECN | - | - | - | 82.9% | - | 71.9% | 77.3% | 3 |
| CDE | 50.0% | 54.7% | 50.0% | 75.8% | - | 50.0% | 56.1% | 4 |
| CCO | 50.0% | 54.7% | 50.0% | 75.8% | - | 50.0% | 56.1% | 5 |
| CDC | 50.0% | 54.7% | 50.0% | 75.8% | - | 50.0% | 56.1% | 6 |
| EIU | - | - | - | 59.4% | - | 50.0% | 54.7% | 7 |
| EAN | - | - | - | 59.4% | - | 50.0% | 54.7% | 8 |
| ERR | 40.0% | 49.7% | 58.0% | 73.2% | - | 40.0% | 52.2% | 9 |
| AIE | 41.8% | 54.5% | 50.8% | - | - | 50.0% | 49.3% | 10 |
| ASF | 29.1% | 11.3% | 24.6% | 88.6% | 0.0% | 42.9% | 32.7% | 11 |
| AEL | 30.9% | 32.0% | 48.1% | 30.5% | 0.0% | 32.3% | 29.0% | 12 |
| CFA | 30.9% | 32.0% | 27.9% | 12.5% | 0.0% | 32.3% | 22.6% | 13 |
| EIN | 0.0% | 9.3% | 24.6% | 57.7% | - | 0.0% | 18.3% | 14 |
| ASI | 14.6% | 14.5% | 22.9% | 0.0% | 0.0% | 18.7% | 11.7% | 15 |
| ACI | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 0.0% | 16 |

使用一组实验来验证MuPri技术得到的变异算子优先级的有效性。主要思路是使用MuPri技术给测试用例集进行排序；计算排序后用例集的APFD值，来反映用例排序的效果；从而评估MuPri技术有效性。

1. **实验步骤**

图5-4给出具体的实验流程，实验步骤描述如下：

1. 首先，按照第3.3.2节的一般过程步骤，根据MuPri技术得到面向BPEL程序*P*的、变异得分为100%且具有一定顺序的测试用例队列*TS*；相应步骤对应图5-4的第一部分；
2. 令*i*=*1，sum*=*0*；
3. 使用μBPEL工具生成程序*P*的变异体集合*R(P)={R1,…, Ri , …Rm}*，*m*为变异体的总数；
4. 依次取出*TS*中的测试用例执行*Ri*，记录第一个将变异体“杀死”的测试用例在*TS*中的位置*s*；
5. 令*sum*=*sum*+*s*；判断*i*是否等于*m*，如果不是，则令*i*=*i*+*1*，返回上一步；否则，执行下一步；
6. 计算测试用例集*TS*的APFD的值。



图5-4 实验流程

1. **实验结果**

以SupplyChain实例为例阐述具体实验过程。对于SupplyChain实例，由表5-2可知，该实例可以应用11种变异算子。由表5-14得到变异算子优先级序列为：CDE->CCO->CDC->ERR->AIE->ASF->AEL->CFA->EIN->ASI->

ACI。根据这个序列，按照图5-4实验流程的第一部分，得到排序后的测试用例集是*TS:T1->T2->T3*。这部分在3.3.3节已经介绍，这里不再详细讨论。

接下来，按照图5-4实验流程的第二部分进行实验。使用测试用例*TS*去执行没有排序的变异体集合，统计APFD。结果如表5-15所示，其中对勾“√”表示第一个将变异体“杀死”的测试用例，*s*表示用例的在*TS*中的位置，“×”表示该测试用例不能“杀死”变异体，“~”表示这个测试用例没有执行。根据APFD计算公式，得出这个测试用例集*TS*的APFD值是74.5%。

表5-15 变异体测试结果

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Mutant** | | **Test Case** | | | **s** |
| T1 | T2 | T3 |
| ERR | 1 | √ | ~ | ~ | 1 |
| 2 | × | × | √ | 3 |
| 3 | √ | ~ | ~ | 1 |
| 4 | × | √ | ~ | 2 |
| 5 | √ | ~ | ~ | 1 |
| EIN | 1 | √ | ~ | ~ | 1 |
| ACI | 1 | √ | ~ | ~ | 1 |
| ASF | 1 | √ | ~ | ~ | 1 |
| 2 | √ | ~ | ~ | 1 |
| AEL | 1 | √ | ~ | ~ | 1 |
| 2 | √ | ~ | ~ | 1 |
| 3 | √ | ~ | ~ | 1 |
| 4 | √ | ~ | ~ | 1 |
| 5 | √ | ~ | ~ | 1 |
| 6 | × | √ | ~ | 2 |
| 7 | √ | ~ | ~ | 1 |
| AIE | 1 | × | √ | ~ | 2 |
| ASI | 1 | √ | ~ | ~ | 1 |
| 2 | √ | ~ | ~ | 1 |
| 3 | √ | ~ | ~ | 1 |
| 4 | √ | ~ | ~ | 1 |
| CFA | 1 | √ | ~ | ~ | 1 |
| 2 | √ | ~ | ~ | 1 |
| 3 | √ | ~ | ~ | 1 |
| 4 | √ | ~ | ~ | 1 |
| 5 | √ | ~ | ~ | 1 |
| 6 | × | √ | ~ | 2 |
| 7 | √ | ~ | ~ | 1 |
| CDE | 1 | √ | ~ | ~ | 1 |
| 2 | × | √ | ~ | 2 |
| CCO | 1 | √ | ~ | ~ | 1 |
| 2 | × | √ | ~ | 2 |
| CDC | 1 | √ | ~ | ~ | 1 |
| 2 | × | √ | ~ | 2 |

对比实验不使用MuPri技术来为变异算子进行排序，仅按照表2-2中变异算子的顺序，为其生成变异体。然后按照5-2流程，得到的测试用例集顺序为*TS:T1->T3>T2*，APFD值为69.6%。其余5个BPEL实例，采取与SupplyChain实例同样的实验过程，完成这两部分实验。统计实验数据，得到如表5-16所示的实验结果。

表5-16 6个BPEL实例实验的APFD值

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Program name** | **P1** | **P2** | **P3** | **P4** | **P5** | **P6** |
| Priority-method | 74.5% | 80.9% | 94.1% | 71.5% | 50% | 70.2% |
| No-Priority-method | 69.6% | 71.9% | 94.1% | 52.8% | 50% | 61.3% |

表5-16中，“Priority-method”代表使用MuPri优化技术的实验，“No-Priority-method”代表对比实验。从结果可以看出，使用MuPri优化技术，得到的测试用例集合的APFD值都大于或等于对比实验的APFD值。由APFD的定义可知，APFD值越大说明该测试用例集检测错误速率越快。所以，实验结果表明，使用MuPri优化技术来为测试用例集进行排序，将会提高测试用例集的检测错误效率，意味着该技术为变异算子排列的顺序也是有效的。

* 1. 小结

本章使用6个BPEL实例并设计了三组不同的实验，对于第三章提出的变异测试优化技术逐一进行了验证与评估。从实验结果来看，MuSR技术通过分析变异算子的操作，剔除具有包含关系的变异算子，减少将近一半的变异体生成；MuSOM技术通过组合一阶变异体来生成二阶变异体，使得变异体数目减半，并且测试的变异得分也没有大幅度降低；使用MuPri技术为这6个实例中可适用的16种变异算子进行了优先级顺序，实验表明此顺序可以有效地为测试用例集排序，提高测试用例集的检测错误效率。

自己总结：扩展标准的WSDL进行约束标注，增强Web服务行为信息描述。通过事件序列图模型描述服务的动态行为，根据服务行为模型生成Web服务的测试序列，并通过实例验证了该方法在测试用例集和错误检测能力方面的有效性，增强了测试路径生成的自动化程度和测试数据的有效性，从而提高了Web服务测试数据自动生成的质量和效率。

1. 工作总结与展望

变异测试广泛用来评估测试技术的有效性和测试用例的完备性。然而,变异测试中大量变异体带来较高的测试开销，造成测试效率低而难以应用到实际中等问题。BPEL是一种新型的服务组装语言，带来一些新的测试挑战。本文在课题组面向BPEL程序的变异测试研究的基础上，研究了面向BPEL程序的变异测试优化技术，取得的主要成果总结如下：

1. 提出了基于包含关系、变异算子优先级和二阶变异体的三种面向BPEL程序的变异测试优化技术，极大的减少BPEL程序的变异测试开销。
2. 开发了面向BPEL程序的变异测试集成化支持工具μBPEL，支持变异测试的全过程，同时还支持本文提出的优化技术，方便测试人员对于BPEL程序不同需求的变异测试。
3. 采用6个BPEL程序实例验证并评估提出的三种面向BPEL程序的变异测试优化技术的有效性。

工作不足及未来展望：

1. 设计并检验了所提出的变异测试优化技术，需要进一步与其他相关的优化技术进行比较，评估优化技术的精简效率。
2. 使用了6个BPEL程序进行实例验证，然而所选取的程序适用的变异算子种类较少，需要开发其他的实例程序进行评估优化技术的有效性。

参考文献

1. M. P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann. Service-Oriented Computing: a Research Roadmap[J]. International Journal of Cooperative Information Systems, 2008, 17(2): 223-255.
2. E. Christensen, F. Curbera, G. Meredith, S. Weerawarana, Web Services Description Language (WSDL)[S], https://www.w3.org/TR/wsdl. A. T. Endo, M. Linschulte, A. D. S. Simão, et al. Event-and Coverage-Based Testing of Web Services, in: Proceedings of the 4th International Conference on Secure Software Integration & Reliability Improvement Companion (SSIRI 2010), IEEE Computer Society, 2010, pp. 62-69.
3. 骆翔宇, 谭征, 苏开乐. 一种基于认知模型检测的Web服务组合验证方法 [J]. 计算机学报, 2011, 34(6): 1041-1061
4. N. Milanovic, M. Malek. Current solutions for web service composition[J]. IEEE Internet Computing, 2004, 8(6): 51-59.
5. F. Belli, A. T. Endo, M. Linschulte, A. Simao. Model-based Testing of Web Service Compositions[C], Proceedings of the 6th IEEE International Symposium on Service Oriented System Engineering (SOSE 2011), IEEE Computer Society, 2011, pp. 181-192.
6. G. Canfora, M. D. Penta. Testing services and service-centric systems: challenges and opportunities[J]. It Professional, 2006, 8(2): 10-17.
7. 姜瑛, 辛国茂, 单锦辉,等. 一种基于合约式设计的测试技术研究[J]. 软件学报, 2004, 15(Suppl): 130-137.
8. C. Peltz. Web services orchestration: A review of emerging technologies, tools, and standards[R]. Technical Report, Hewlett-Packard Company, http://devresource.hp.com/drc/
9. C. Ma, C. Du, T. Zhang, et al. WSDL-Based Automated Test Data Generation for Web Service[C] Proceedings of the 2008 International Conference on Computer Science and Software Engineering (CSSE 2008), Wuhan, China, 2008:731-737.
10. OASIS Standard. Web services business process execution language version 2.0[EB/OL].http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html, 2012.
11. 宋波, 李妙妍. 面向Web服务的BPEL的研究与实现 [J]. 计算机工程与实现, 2007, 28(9): 2212-2214.
12. DeMillo, R. A., Lipton, R. J., Sayward, F. G. Hints on test data selection: Help for the practicing programmer [J]. IEEE Computer, 1978, 11(4): 34-41.
13. Estero-Botaro, A., Palomo-Lozano, F., and Medina-Bulo, I. Mutation Operators for WS-BPEL 2.0 [C]//Proceedings of the International Conferen -ce on Software and Systems Engineering and their Applications (ICSSEA’08), 2008: 1-7.
14. Estero-Botaro, A., Palomo-Lozano, F., Medina-Bulo, I. Quantitative Evaluation of Mutation Operators for WS-BPEL Compositions [C]// Proceedings of 3rd International Conference on Software Testing, Verification, and Validation Workshops (ICSTW), IEEE Computer Society, 2010: 142-150.
15. 王巧玲. 面向BPEL程序的变异测试技术与支持工具研究 [D]. 北京科技大学, 2015.
16. Sun. C.-A., Pan. L., Wang. Q. L., et al. An Empirical Study on Mutation Testing of WS-BPEL Programs [J]. The Computer Journal, 2016, in press (DOI:10.1093/comjnl/bxw076).
17. Zhang, L., Hou, S.-S., Hu, J.-J., Xie, T., and Mei, H. Is operator-based mutant selection superior to random mutant selection? [C]//Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering (ICSE'10), 2010: 435-444.
18. Sun, C.-A., Zhai, Y. M., Zhang, Z. BPELDebugger: An effective BPEL -specific fault localization framework [J]. Information and Software Technology, 2013, 55(12): 2140-2153.
19. 陈翔，顾庆. 变异测试：原理、优化和应用 [J]. 计算机科学与探索,2012, 6(12): 1057-1075.
20. Jia, Y., Harman, M. An analysis and survey of the development of mutation testing [J]. IEEE Transactions on Software Engineering, 2011, 37(5): 649-6 78.
21. Fraser, G., Zeller, A. Mutation-driven generation of unit tests and oracles [C]//Proceedings of the19th International Symposium on Software Testing and Analysis (ISSTA'10), 2010: 147-158.
22. Appelt, D., Nguyen, C. D., Briand, L. C., et al. Automated testing for SQL injection vulnerabilities: an input mutation approach [C]//Proceedings of the 2014 International Symposium on Software Testing and Analysis. ACM, 2014: 259-269.
23. Gopinath, R., Jensen, C., Groce, A. Code coverage for suite evaluation by developers [C]//Proceedings of the 36th International Conference on Software Engineering (ICSE'14), 2014: 72-82.
24. Inozemtseva, L., Holmes, R. Coverage is not strongly correlated with test suite effectiveness [C]//Proceedings of the 36th International Conference on Software Engineering (ICSE'14), 2014: 435-445.
25. Chen, T. Y., Kuo, F.-C., Liu, H., Wong, W. E. Code coverage of adaptive random testing [J]. IEEE Transactions on Reliability, 2013, 62(1): 226-237.
26. Liu, H., Kuo, F.-C., Towey, D., Chen, T. Y. How effectively does metamorphic testing alleviate the oracle problem? [J]. IEEE Transactions on Software Engineering, 2014, 40(1): 4-22.
27. Wright, C. J., Kapfhammer, G. M., and McMinn, P. The impact of equivalent, redundant and quasi mutants on database schema mutation analysis [C]//Proceedings of the 14th International Conference on Quality Software (QSIC'14), 2014:57-66.
28. Zhang, L., Marinov, D., Zhang, L., and Khurshid, S. Regression mutation testing [C]//Proceedings of the 2012 International Symposium on Software Testing and Analysis (ISSTA'12), 2012: 331-341.
29. Sun, C.-A., Zhao, Y., Pan, L., Liu, H., and Chen, T. Y. Automated Testing of WS-BPEL Service Compositions: a Scenario-oriented Approach [J]. IEEE Transactions on Services Computing, 2015, in press (DOI: 10.1109 /TSC.2015.2466572).
30. Sun, C.-A., Wang, G., Cai, K.-Y., Chen, T. Y. Distribution-aware mutation analysis [C]//Proceedings of 9th IEEE International Workshop on Software Cybernetics (IWSC'12), 2012: 170-175.
31. Just, R., Jalali, D., Nozemtseva, L., Ernst, I M. D., Holmes, R., Fraser, G. Are mutants a valid substitute for real faults in software testing? [C] //Proceedings of the Symposium on the Foundations of Software Engineering (FSE2014), 2014: 654-665.
32. García-Domínguez, A., Medina-Bulo, I. MuBPEL [EB/OL]. https:// neptuno.uca.es/redmine/projects/sources-fm/wiki/MuBPEL.html, 2015.
33. Mathur, A. P., Wong, W. E. An empirical comparison of data flow and mutation-based test adequacy criteria [J]. Software Testing, Verification and Reliability, 1994, 4(1):9-31.
34. Hussain, S. Mutation clustering [D]. London, UK: King’s College, 2008.
35. King, K. N., Offutt, A. J. A FORTRAN language system for mutation based software testing [J]. Software Testing, Verification and Reliability, 1991, 21(7): 685-718.
36. Offutt, A. J., Rothermel, G., Zapf, C. An experimental evaluation of selective mutation [C]//Proceedings of the 15th International Conference on Software Engineering (ICSE ’93), 1993: 100-107.
37. Langdon, W. B., Harnab, N., Jia, Y. Efficient multi-objective higher order mutation testing with genetic programming [J]. Journal of Systems and Software, 2010, 83(12): 2416-2430.
38. 薛飞飞. 基于控制流的变异体精简技术与支持工具研究 [D]. 北京科技大学, 2016.
39. Sun, C.-A., Xue, F. F., Liu, H., Zhang, X. Y. A Path-aware Approach to Mutant Reduction in Mutation Testing, Information and Software Technology, Elsevier, 2016, 81(1): 65-81.
40. Krauser, E. W., Mathur, A. P., Rego, V. J. High performance software testing on SIMD machines [J]. IEEE Transactions on Software Engineering, 1991, 17(5): 403-423.
41. 牟小玲. 基于扩展着色Petri网的服务组合测试研究 [D]. 西南大学, 2012.
42. LEE, S., OFFUTT, J. Generating test cases for XML-based Web component interactions using mutation analysis [C]//Proceedings of the 12th Internation -al Symposium on Software Reliability Engineering (ISSRE ’01), IEEE Computer Society, 2001: 200-209.
43. Boonyakulsrirung, P., Suwannasart, T. A weak mutation testing framework for WS-BPEL [C]// Proceedings of the 8th International Joint Conference on Computer Science and Software Engineering (JCSSE'11), IEEE, 2011: 313 -318.
44. 赵彦. 基于场景的BPEL测试用例自动生成技术与工具研究 [D]. 北京科技大学, 2014.
45. Domínguez-Jiménez, J. J., Estero-Botaro, A., García-Domínguez, A., et al. GAmera: an automatic mutant generation system for WS-BPEL compositions [C]//Proceedings of the 7th European Conference on Web Services (ECOWS'09), IEEE, 2009: 97-106.
46. Boonyakulsrirung, P., Suwannasart, T. WeMuTe - a weak mutation testing tool for WS-BPEL [C]//Proceedings of The International Multi Conference of Engineers and Computer Scientists (IMECS'12), 2012: 810-815.
47. Polo, M., Piattini, M., García-Rodríguez, I. Decreasing the cost of mutation testing with second-order mutants [J]. Software Testing, Verification and Reliability, 2009, 19(2): 111-131.
48. dom4j 2.0 [EB/OL]. http://dom4j.sourceforge.net/, 2010.
49. Active VOS. Car repair estimation [EB/OL]. http://www.activevos.com/ developers/sample-apps, 2012.
50. Sun, C.-A., Khoury, E., Aiello, M. Transaction management in service -oriented systems: Requirements and a proposal [J]. IEEE Transactions on Services Computing, 2011, 4: 167-180.
51. Elbaum, S., Malishevs, A. G., Rothermel, G. Test case prioritization: A family of empirical studies [J]. IEEE Transactions on Software Engineering, 2002, 28(10): 159-182.
52. 陈梦云. 基于圈复杂度和调用次数的测试用例排序方法 [D]. 上海师范大学, 2015.
53. Myers, G. J. The Art of Software Testing, second edition [M]. John Wiley and Sons, 2004.

作者简历及在学研究成果

1. 作者入学前简历

|  |  |  |
| --- | --- | --- |
| 起止年月 | 学习或工作单位 | 备注 |
| 2010年09月至2014年06月 | 在北京科技大学计算机科学与技术专业攻读 |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. 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.

[2] 高效新型的SOA 软件测试技术与工具研究, 北京市优秀人才培养项目（2012D009006000002），主要研究人员.

1. 在学期间所获的科研奖励

[1] μBPEL：一个面向BPEL服务组装程序的变异测试系统, 全国软件原型竞赛二等奖, 中国计算机学会软件工程专业委员会与中国计算机学会系统软件专业委员会，2015.11.

[2] 研究生国家奖学金，北京科技大学，2016.10.

1. 在学期间发表的论文

[1] Sun, C.-A., Pan, L., Wang, Q. L., et al. An empirical study on mutation testing of WS-BPEL programs [J]. The Computer Journal, 2016.（已发表. SCI刊源）

[2] Sun, C.-A., Zhao, Y., Pan, L., et al. Automated testing of WS-BPEL service compositions-A scenario - oriented approach [J]. IEEE Transactions on Services Computing, 2015.（已发表. SCI刊源）

[3] Sun, C.-A., Zhao, Y., Pan, L., et al. A transformation-based approach to testing concurrent programs using UML activity diagrams [J]. Software: Practice and Experience, 2016, 46(4): 551-576.（已发表. SCI刊源. SCI检索号000372324600005）

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** |
| 服务组装，BPEL，变异测试，测试工具，包含关系 | 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\*** |
| BPEL程序的变异测试优化技术与集成化支持工具研究 | |  | | 中文 |
| **作者姓名\*** | 潘琳 | | **学号\*** | G20148605 |
| **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\*** |
| 计算机技术 | | 软件测试 | 2.5年 | 2017 |
| **论文提交日期\*** | 2016年12月21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** | |
| 王昭顺  施小丁 | 王昭顺 | | 罗熊、王卫苹 | |
| **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\*** | 59页 | | | |
| 共33项，其中带\*为必填数据，为22项。 | | | | |

1. [] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann. Service-Oriented Computing: a Research Roadmap[J]. International Journal of Cooperative Information Systems, 2008, 17(2): 223-255. [↑](#endnote-ref-1)
2. [] E. Christensen, F. Curbera, G. Meredith, S. Weerawarana. Web Services Description Language (WSDL)[S]. <https://www.w3.org/TR/wsdl>. [↑](#endnote-ref-2)
3. [] 骆翔宇, 谭征, 苏开乐. 一种基于认知模型检测的Web服务组合验证方法 [J]. 计算机学报, 2011, 34(6): 1041-1061. [↑](#endnote-ref-3)
4. [] N. Milanovic, M. Malek. Current solutions for web service composition[J]. IEEE Internet Computing, 2004, 8(6): 51-59. [↑](#endnote-ref-4)
5. [] F. Belli, A.T. Endo, M. Linschulte, A. Simao. Model-based Testing of Web Service Compositions[C]. Proceedings of the 6th IEEE International Symposium on Service Oriented System Engineering (SOSE 2011), IEEE Computer Society, 2011, pp. 181-192. [↑](#endnote-ref-5)
6. [] G. Canfora, M.D. Penta. Testing services and service-centric systems: challenges and opportunities[J]. It Professional, 2006, 8(2): 10-17. [↑](#endnote-ref-6)
7. [] 姜瑛, 辛国茂, 单锦辉,等. 一种基于合约式设计的测试技术研究[J]. 软件学报, 2004, 15(Suppl): 130-137. [↑](#endnote-ref-7)
8. [] C. Peltz. Web services orchestration: A review of emerging technologies, tools, and standards[R]. Technical Report, Hewlett-Packard Company, <http://devresource.hp.com/drc/>. [↑](#endnote-ref-8)
9. [] C. Ma, C. Du, T. Zhang, et al. WSDL-Based Automated Test Data Generation for Web Service[C]. Proceedings of the 2008 International Conference on Computer Science and Software Engineering (CSSE 2008), Wuhan, China, 2008, pp. 731-737. [↑](#endnote-ref-9)
10. [] 缪淮扣, 陈圣波, 曾红卫. 基于模型的Web应用测试[J]. 计算机学报, 2011, 34(6): 1012-1028. [↑](#endnote-ref-10)
11. [] Zoltán Micskei. Model-based testing (MBT)[DB/OL]. <http://mit.bme.hu/~micskeiz/pages/mbt.html>, 2017-09-25. [↑](#endnote-ref-11)
12. [] S.R. Dalal, A. Jain, N. Karunanithi, B.M. Horowitz. Model-Based Testing in Practice[C]. Proceedings of the 21th International Conference on Software Engineering (ICSE 1999), ACM Press, 1999, pp. 285-294. [↑](#endnote-ref-12)
13. [] A.C.D. Neto, R. Subramanyan, M. Vieira, et al. A Survey on Model-based Testing Approaches: A Systematic Review[C]. Proceedings of the 1st ACM International Workshop on Empirical Assessment of Software Engineering Languages and Technologies (WEASETech 2007), ACM, 2007, pp. 31-36. [↑](#endnote-ref-13)
14. [] M. Utting, A. Pretschner, B. Legeard. A taxonomy of model-based testing approaches[J]. Software Testing Verification & Reliability, 2012, 22(5): 297–312. [↑](#endnote-ref-14)
15. [] K. Kristian. GraphWalker[DB/OL]. <http://graphwalker.github.io>, 2017-09-15. [↑](#endnote-ref-15)
16. [] J. Ellson, E. Gansner, E. Koutsofios, et al. Graphviz-Open Source Graph Drawing Tools[J]. Lecture Notes in Computer Science, 2001, 2265: 483-484. [↑](#endnote-ref-16)
17. [] J. Ellson, E.R. Gansner, E. Koutsofios, et al. Graphviz and Dynagraph - Static and Dynamic Graph Drawing Tools[M]. Graph Drawing Software. Springer Berlin Heidelberg, 2004: 127-148. [↑](#endnote-ref-17)
18. [] E.R. Gansnerh, E. Koutsofios, S. North. Graphviz - Graph Visualization Software [DB/OL]. <http://www.graphviz.org/Home.php>. [↑](#endnote-ref-18)
19. [] E.R. Gansnerh, E. Koutsofios, S. North. Drawing graphs with dot[DB/OL]. <http://www.graphviz.org/pdf/dotguide.pdf>, 2015-01-05. [↑](#endnote-ref-19)
20. [] 王翀, 吕荫润, 陈力,等. SMT求解技术的发展及最新应用研究综述[J]. 计算机研究与发展, 2017, 54(7): 1405-1425. [↑](#endnote-ref-20)
21. [] Leonardo de Moura and Nikolaj Bjørner. Z3:An Efficient SMT Solver[C]. Proceedings of the 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2008), Springer, 2008, pp. 337-340. [↑](#endnote-ref-21)
22. [] Y. Zheng, X. Zhang, V. Ganesh. Z3-str: A Z3-Based String Solver for Web Application Analysis[C]. Proceedings of the Joint Meeting of the 14th European Software Engineering Conference and the 21st ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2013), ACM, 2013, pp. 114-124. [↑](#endnote-ref-22)
23. [] L. Cordeiro, B. Fischer, J. Marquessilva. SMT-Based Bounded Model Checking for Embedded ANSI-C Software[J]. IEEE Transactions on Software Engineering, 2012, 38(4): 957-974. [↑](#endnote-ref-23)
24. [] A. Kiezun, V. Ganesh, P.J. Guo, et al. HAMPI: A Solver for String Constraints[C]. Proceedings of the 18th International Symposium on Software Testing and Analysis (ISSTA 2009), ACM, 2009, pp. 105-116. [↑](#endnote-ref-24)
25. [] P. Bisht, T. Hinrichs, N. Skrupsky, et al. NoTamper: Automatic Blackbox Detection of Parameter Tampering Opportunities in Web Applications[C]. Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS 2010), ACM, 2010, PP. 607-618. [↑](#endnote-ref-25)
26. [] W.T. Tsai, R. Paul, Y. Wang, et al. Extending WSDL to Facilitate Web Services Testing[C]. Proceedings of the 7th IEEE International Symposium on High-Assurance Systems Engineering (HASE 2002), IEEE Computer Society, 2002, pp. 171-172. [↑](#endnote-ref-26)
27. [] Q.Z. Sheng, Z. Maamar, L. Yao, et al. Behavior modeling and automated verification of Web services[J]. Information Sciences, 2014, 258(3): 416-433. [↑](#endnote-ref-27)
28. [] H.M. Sneed, S. Huang. WSDLTest - A Tool for Testing Web Services[C]. Proceedings of the 8th IEEE International Workshop on Web Site Evolution (WSE 2006). IEEE Computer Society, 2006, pp. 14-21. [↑](#endnote-ref-28)
29. [] A. Bertolino, A. Polini. The Audition Framework for TestingWeb Services Interoperability[C]. Proceedings of the 31st EUROMICRO Conference on Software Engineering and Advanced Applications (EUROMICRO-SEAA 2005). IEEE Computer Society, 2005, pp. 134-142. [↑](#endnote-ref-29)
30. [] R. Heckel, L. Mariani. Automatic Conformance Testing of Web Services[C]. Proceedings of the 8th International Conference on Fundamental Approaches to Software Engineering (FASE 2005), Springer-Verlag, 2005, pp. 34-48. [↑](#endnote-ref-30)
31. [] N. Parimala, A. Saini. Web Service with Criteria: Extending WSDL[C]. Proceedings of the 6th IEEE International Conference on Digital Information Management (ICDIM 2011), IEEE, 2011, pp. 205-210. [↑](#endnote-ref-31)
32. [] L. Jiang, T. Liu, D. Liu. Objective and SUbjective QoS factors supported Web Service search method based on extended WSDL[C]. Proceedings of the 23rd International Conference on Geoinformatics (Geoinformatics 2015), IEEE, 2015, pp. 1-4. [↑](#endnote-ref-32)
33. [] P.W. Wang, Z.J. Ding, C.J. Jiang, et al. Constraint-Aware Approach to Web Service Composition[J]. IEEE Transactions on Systems Man & Cybernetics Systems, 2017, 44(6):770-784. [↑](#endnote-ref-33)
34. [] 袁雪莉. 基于扩展WSDL的测试用例自动生成[D]. 重庆:西南大学, 2009. [↑](#endnote-ref-34)
35. [] 张峻. 基于UML2.0动态视图的Web服务模型测试方法及其应用[D]. 苏州大学, 2007. [↑](#endnote-ref-35)
36. [] Y. Zheng, J. Zhou, P. Krause. An Automatic Test Case Generation Framework for Web Services[J]. Journal of Software, 2007, 2(3): 64-77. [↑](#endnote-ref-36)
37. [] A. Bertolino, G. D. Angelis, L. Frantzen, A. Polini. Model-Based Generation of Testbeds for Web Services[C]. Proceedings of the 20th IFIP International Conference on Testing of Software and Communicating Systems (TESTCOM 2008), Springer, 2008, pp. 266-282. [↑](#endnote-ref-37)
38. [] A. T. Endo, A. Simao. Model-Based Testing of Service-Oriented Applications via State Models[C]. Proceedings of the 8th IEEE International Conference on Services Computing (SCC 2011), IEEE Computer Society, 2011, pp. 432-439. [↑](#endnote-ref-38)
39. [] C. S. Keum, S. Kang, I. Y. Ko, J. Baik, Y. I. Choi. Generating Test Cases for Web Services Using Extended Finite State Machine[C]. Proceedings of the 18th IFIP International Conference on Testing of Software and Communicating Systems (TESTCOM 2006), Springer, 2006, pp. 103-117. [↑](#endnote-ref-39)
40. [] A. S. Kalaji, R. M. Hierons, S. Swift. An integrated search-based approach for automatic testing from extended finite state machine (EFSM) models[J]. Information & Software Technology, 2011, 53(53): 1297-1318. [↑](#endnote-ref-40)
41. [] M. Kiran, A. J. H. Simons. Model-Based Testing for Composite Web Services in Cloud Brokerage Scenarios[C]. Proceedings of the 3rd European Conference on Service-Oriented and Cloud Computing (ESOCC 2014), Springer, 2014, pp. 190-205. [↑](#endnote-ref-41)
42. [] A. T. Endo, M. Linschulte, A. D. S. Simão, et al. Event-and Coverage-Based Testing of Web Services[C]. Proceedings of the 4th International Conference on Secure Software Integration & Reliability Improvement Companion (SSIRI 2010), IEEE Computer Society, 2010, pp. 62-69. [↑](#endnote-ref-42)
43. [] F. Belli, M. Linschulte. Event-Driven Modeling and Testing of Web Services[C]. Proceedings of the 32nd IEEE International Computer Software and Applications Conference (COMPSAC 2008), IEEE Computer Society, 2008, pp. 1168-1173. [↑](#endnote-ref-43)
44. [] F. Belli, A. T. Endo, M. Linschulte, A. Simao. A holistic approach to model-based testing of Web service compositions[J]. Software Practice & Experience, 2014, 44(2): 201–234. [↑](#endnote-ref-44)
45. [] C. S. Wu, C. H. Huang. The Web Services Composition Testing Based on Extended Finite State Machine and UML Model[C]. Proceedings of the 5th International Conference on Service Science and Innovation (ICSSI 2013), IEEE Computer Society, 2013, pp. 215-222. [↑](#endnote-ref-45)
46. [] G. Zhang, R. Mei, J. Zhang. A Business Process of Web Services Testing Method Based on UML2.0 Activity Diagram[C]. Proceedings of the Workshop on Intelligent Information Technology Application (IITA 2007), IEEE Computer Society, 2007, pp. 59-65. [↑](#endnote-ref-46)
47. [] V. Pretre, A. D. Kermadec, F. Bouquet, et al. Automated UML models merging for web services testing[J]. International Journal of Web & Grid Services, 2009, 5(2): 107-129. [↑](#endnote-ref-47)
48. [] S. Ali, L. C. Briand, J. U. Rehman, et al. A state-based approach to integration testing based on UML models[J]. Information & Software Technology, 2007, 49(11–12): 1087-1106. [↑](#endnote-ref-48)
49. [] M. E. Cambronero, G. Diaz, J. J. Pardo, V. Valero. Using UML Diagrams to Model Real-Time Web Services[C]. Proceedings of the 2nd International Conference on Internet and Web Applications and Services (ICIW 2007), IEEE Computer Society, 2007, pp. 24. [↑](#endnote-ref-49)
50. [] D. Lee, M. Yannakakis. Principles and methods of testing finite state machines-a survey[J]. Proceedings of the IEEE, 1996, 84(8): 1090-1123. [↑](#endnote-ref-50)
51. [] F. Belli, A. Hollmann, S. Padberg. Model-Based Integration Testing with Communication Sequence Graphs. In Model-Based Testing for Embedded Systems, CRC Press, 2011, pp. 223-243. [↑](#endnote-ref-51)
52. [] 韩德帅, 杨启亮, 邢建春. 一种软件自适应UML建模及其形式化验证方法[J]. 软件学报, 2015, 26(4): 730-746. [↑](#endnote-ref-52)
53. [] C. Sun, G. Wang, B. Mu, H. Liu, Z. Wang, T. Y. Chen. Metamorphic Testing for Web Services: Framework and a Case Study[C]. Proceedings of the 9th IEEE International Conference on Web Services (ICWS 2011), IEEE Computer Society, 2011, pp. 283-290. [↑](#endnote-ref-53)
54. [] C. 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 on Web Service Research, 2012, 9(1): 51-73. [↑](#endnote-ref-54)
55. [] C. Sun, G. Wang, K. Y. Cai, T. Y. Chen. Towards Dynamic Random Testing for Web Services[C]. Proceedings of the 36th Annual IEEE International Computer Software and Applications Conference (COMPSAC 2012), IEEE Computer Society, 2012, pp. 164-169. [↑](#endnote-ref-55)
56. [] C. Sun, Y. Zhai, Y. Shang, Z. Zhang. BPELDebugger: An effective BPEL-specific fault localization framework[J]. Information and Software Technology, 2013, 55(12): 2140-2153. [↑](#endnote-ref-56)
57. [] C. Peltz. Web services orchestration and choreography[J]. IEEE Computer, 2003, 36(10): 46-52. [↑](#endnote-ref-57)
58. [] C. Sun, Y. Shang, Y. Zhao, T. Y. Chen. Scenario-Oriented Testing for Web Service Compositions Using BPEL[C]. Proceedings of the 12th International Conference on Quality Software (QSIC 2012), IEEE Computer Society, 2012, pp. 171-174. [↑](#endnote-ref-58)
59. [] C. Sun, Y. Zhao, L. Pan, L. Hui, T. Y. Chen. Automated Testing of WS-BPEL Service Compositions: A Scenario-Oriented Approach[J]. IEEE Transactions on Services Computing, in press (accepted on 2 August 2015). [↑](#endnote-ref-59)
60. [] 侯可佳, 白晓颖, 陆皓,等. 基于接口语义契约的Web服务测试数据生成[J]. 软件学报, 2013, 24(9): 2020-2041. [↑](#endnote-ref-60)
61. [] 许蕾, 陈林, 徐宝文. 用户需求驱动的Web服务测试[J]. 计算机学报, 2011, 34(6): 1029-1040. [↑](#endnote-ref-61)
62. [] 李盛钢. 一种基于扩展WSDL的测试数据自动生成方法[D]. 西南大学, 2010. [↑](#endnote-ref-62)
63. [] P. Bourhis, J.L. Reutter. JSON: Data model, Query languages and Schema specification[C]. Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS 2017), ACM, 2017, pp. 123-135. [↑](#endnote-ref-63)
64. [] F. Belli, C.J. Budnik, et al. Event-based modelling, analysis and testing of user interactions: approach and case study: Research Articles[J]. Software Testing Verification & Reliability, 2006, 16(1): 3–32. [↑](#endnote-ref-64)
65. [] C. Sun. A Transformation-based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications[C]. Proceedings of 32nd Annual IEEE International Computer Software and Application Conference (COMPSAC 2008)，IEEE Computer Society, Turku, Finland, 2008, pp. 160-167. [↑](#endnote-ref-65)