面向BPEL程序的变异测试技术与支持工具研究

王巧玲

北京科技大学

**密　　　　级：\_\_\_\_\_\_\_\_\_\_\_\_\_**

论文题目：面向BPEL程序的变异测试技术

与支持工具研究

S20121223

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

王巧玲

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

计算机科学与技术

专业名称：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2014年12月22日

面向BPEL程序的变异测试技术与支持工具研究

Research on Mutation Testing Technique for BPEL Programs and Its Supporting Tool

研究生姓名：王巧玲

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：Wang Qiaoling

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**

面向BPEL程序的变异测试技术与支持工具研究

**论文题目：**

王巧玲

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

孙昌爱

北京科技大学

**指 导 教 师： 单位：**

**指导小组成员： 单位：**

**单位：**

**论文提交日期：**2014年 12月 22日

**学位授予单位：北 京 科 技 大 学**

致谢

衷心感谢我的导师孙昌爱老师在我三年来的学习生涯里给我的帮助和细心的指导。不论是我在大四毕设期间，还是在我两年半的研究生学习期间，都给了我悉心的指导，尤其感谢老师在研一期间对我懒散态度的教诲。孙老师在我研究生期间给我指明了方向，不管是在硕士期间的课题研究项目，还是在论文的选题、研究以及修改过程中，孙老师都认真负责指导我完成研究任务。孙老师不仅在学习上指导我的研究，在生活中也是一位积极乐观、认真负责的人，教导了很多做人做事的道理。

感谢实验室同学们的无私帮助，感谢尚岩师兄和赵彦师姐的指导，感谢徐晓巍同学和徐良政同学对系统的前期的贡献。

感谢我的父母亲人们，始终在物质和精神上无怨无悔的支持我，

最后，向在百忙之中抽出宝贵时间评阅本论文的老师们致以诚挚的谢意。

摘要

BPEL是一种基于XML的可执行的服务组合语言。BPEL程序通过协调多个松散、分布的Web服务完成复杂的业务流程。与传统的应用程序相比，BPEL程序呈现出许多新的特征。为了确保BPEL程序的正确性，有必要对其进行系统、高效测试。变异测试是一种基于故障的测试技术，具有很强的故障检测能力。

本文探讨将变异测试应用于新型的BPEL程序时的关键问题，提出了面向BPEL程序的变异测试技术，设计面向BPEL程序的变异测试框架，开发相应的支持工具。本文的主要成果如下：

* **提出面向BPEL程序的变异测试框架：**根据测试人员提供的BPEL程序自动生成变异体，根据变异体对BPEL程序进行变异测试。
* **开发面向BPEL程序的变异测试系统MuBPEL：**实现了BPEL变异体的自动生成，对BPEL程序进行变异测试，能够对变异测试结果进行分析，提高了BPEL程序的变异测试的自动化程度。
* **经验研究：**采用六个BPEL程序实例验证了开发的变异测试框架与工具的有效性；分析与评估了不同类型变异算子产生的BPEL程序变体检测的容易程度，为BPEL程序变异测试精简变异算子提供了依据。

面向BPEL程序的变异测试技术提供了一种有效的BPEL程序测试手段，不同BPEL变异算子的故障检测关系有助于精简变异算子，进一步提高了变异测试的效率。借助本文开发的面向BPEL程序的变异测试技术与工具，测试人员不仅可以对BPEL程序进行高效的变异测试，还可用来评估现有面向BPEL程序的测试技术的有效性。

关键词： BPEL，变异测试，变异算子，MuBPEL，精简技术

Research on Mutation Testing Technique for BPEL Programs and Its Supporting Tool

Abstract

BPEL is an XML-based service composition language which can be used to define a complex business process by orchestrating multiple Web services. A BPEL program is made up of several loosely-coupled Web services. Compared with traditional applications, BPEL program presents many new features. In order to verify the correctness of the BPEL programs, it is necessary to carry out systematic and efficient testing. Mutation testing is a kind of fault-based software testing techniques with a strong fault detection capability.

This thesis will explore the key issues of mutation testing when it is applied to BPEL programs. This thesis proposed a mutation testing technique for BPEL programs with a mutation testing framework for BPEL programs, and a corresponding supporting tool. The main contributions made in this thesis are as follows:

* A mutation testing framework for BPEL programs, which is able to automatically generate mutants based on BPEL programs provided by testers, and leverage applicable mutants to execute mutation testing of BPEL programs.
* A mutation testing tool for BPEL programs called MuBPEL, which has been developed to implement automatic BPEL mutant generation, aid the execution testing and verification and analysis of test result. This tool is helpful to improve the efficiency of mutation testing of BPEL programs since it enhances its automation.
* An empirical study where six BPEL programs were used to validate the effectiveness of the presented framework and tool. An evaluation of fault detection of different types of mutation operators is provided, which can serve as a basis for reducing mutation operators for mutation testing of BPEL programs.

The presented mutation testing technique provides a kind of effective BPEL programs testing approach. Evalution of fault detection of different mutation operators is helpful to reduce the mutation operators, and thus further improve the efficiency of mutation testing. With the aid of the mutation testing technique and supporting tool developed in this thesis, testers can not only efficiently exercise mutation testing of BPEL programs, but also evaluate the effectiveness of existing testing techniques for BPEL programs.

Key Words：BPEL, Mutation Testing, Mutation Operator, MuBPEL,

Reduce Technique

目录

[致谢 I](#_Toc406856747)

[摘要 III](#_Toc406856748)

[Abstract V](#_Toc406856749)

[插图和附表清单 IX](#_Toc406856750)

[1 引言 1](#_Toc406856751)

[2 背景介绍 2](#_Toc406856752)

[2.1 相关概念及技术 2](#_Toc406856753)

[2.1.1 BPEL 2](#_Toc406856754)

[2.1.2 变异测试 4](#_Toc406856755)

[2.2 国内外研究现状 5](#_Toc406856756)

[2.3 研究背景与意义 7](#_Toc406856757)

[2.4 研究内容及成果 7](#_Toc406856758)

[2.5 论文组织结构 8](#_Toc406856759)

[3 面向BPEL程序的变异测试 9](#_Toc406856760)

[3.1 面向BPEL程序的变异测试框架 9](#_Toc406856761)

[3.2 面向BPEL程序的变异测试过程 10](#_Toc406856762)

[3.3 示例 11](#_Toc406856763)

[4 面向BPEL程序的变异测试系统设计与实现 12](#_Toc406856764)

[4.1 需求分析 12](#_Toc406856765)

[4.2 系统架构 14](#_Toc406856766)

[4.3 系统设计 16](#_Toc406856767)

[4.3.1 变异体自动生成子系统设计 16](#_Toc406856768)

[4.3.2 变异测试执行子系统设计 20](#_Toc406856769)

[4.3.3 测试结果分析子系统设计 21](#_Toc406856770)

[4.4 系统实现与演示 22](#_Toc406856771)

[4.4.1 系统功能 22](#_Toc406856772)

[4.4.2 系统演示 22](#_Toc406856773)

[4.5 小结 28](#_Toc406856774)

[5 经验研究 29](#_Toc406856775)

[5.1 实例SupplyChain的变异测试 29](#_Toc406856776)

[5.2 实例SmartShelf的变异测试 32](#_Toc406856777)

[5.3 实例SupplyCustomer的变异测试 37](#_Toc406856778)

[5.4 实例loan\_approval的变异测试 41](#_Toc406856779)

[5.5 实例CarEstimate的变异测试 44](#_Toc406856780)

[5.6 实例TravelAgency的变异测试 46](#_Toc406856781)

[5.7 变异体精简 50](#_Toc406856782)

[5.8 小结 53](#_Toc406856783)

[6工作总结与展望 54](#_Toc406856784)

[参考文献 55](#_Toc406856785)

[附录 面向BPEL2.0的变异算子 59](#_Toc406856786)

[作者简历及在学研究成果 60](#_Toc406856787)

[独创性说明 61](#_Toc406856788)

[关于论文使用授权的说明 61](#_Toc406856789)

[学位论文数据集 1](#_Toc406856790)

插图和附表清单

[图2-1 BPEL基本结构 3](#_Toc406946709)

[图2-2 BPEL活动示例 3](#_Toc406946710)

[图3-1 面向BPEL程序的变异测试框架图 10](#_Toc406946711)

[图4-1 MuBPEL的用例图 13](#_Toc406946712)

[图4-2 MuBPEL的系统架构图 15](#_Toc406946713)

[图4-3 变异体自动生成时序图 16](#_Toc406946714)

[图4-4 变异测试执行时序图 20](#_Toc406946715)

[图4-5 BPEL原始程序选择界面 23](#_Toc406946716)

[图4-6 BPEL原始程序查看界面 23](#_Toc406946717)

[图4-7 BPEL变异算子选择界面 24](#_Toc406946718)

[图4-8 BPEL变异体生成界面 24](#_Toc406946719)

[图4-9 MuBPEL变异测试界面 25](#_Toc406946720)

[图4-10 MuBPEL测试用例读取界面 25](#_Toc406946721)

[图4-11 MuBPEL原始程序测试界面 26](#_Toc406946722)

[图4-12 MuBPEL变异体测试界面 27](#_Toc406946723)

[图4-13 MuBPEL变异体报告界面 27](#_Toc406946724)

[图5-1 SupplyChain执行流程 30](#_Toc406946725)

[图5-2 SmartShelf执行流程 33](#_Toc406946726)

[图5-3 SupplyCustomer执行流程 38](#_Toc406946727)

[图5-4 loan\_approval执行流程 41](#_Toc406946728)

[图5-5 CarEstimate执行流程 44](#_Toc406946729)

[图5-6 TravelAgency执行流程 47](#_Toc406946730)

[表5-1 SupplyChain的输入输出 30](#_Toc406946731)

[表5-2 SupplyChain的数据库 31](#_Toc406946732)

[表5-3 SupplyChain的变异算子 31](#_Toc406946733)

[表5-4 SupplyChain故障检测率 32](#_Toc406946734)

[表5-5 SupplyChain实验变异得分 32](#_Toc406946735)

[表5-6 SmartShelf的输入输出 34](#_Toc406946736)

[表5-7 SmartShelf的数据库 34](#_Toc406946737)

[表5-8 SmartShelf的变异算子 34](#_Toc406946738)

[表5-9 SmartShelf故障检测率 35](#_Toc406946739)

[表5-9 SmartShelf故障检测率（续） 36](#_Toc406946740)

[表5-10 SmartShelf实验变异得分 37](#_Toc406946741)

[表5-11 SupplyCustomer的输入输出 38](#_Toc406946742)

[表5-12 SupplyCustomer的变异算子 39](#_Toc406946743)

[表5-13 SupplyCustomer故障检测率 40](#_Toc406946744)

[表5-14 SupplyCustomer实验变异得分 41](#_Toc406946745)

[表5-15 loan\_approval的变异算子 42](#_Toc406946746)

[表5-16 loan\_approval故障检测率 43](#_Toc406946747)

[表5-17 loan\_approval实验变异得分 43](#_Toc406946748)

[表5-18 CarEstimate的变异算子 45](#_Toc406946749)

[表5-19 CarEstimate故障检测率 46](#_Toc406946750)

[表5-20 CarEstimate实验变异得分 46](#_Toc406946751)

[表5-21 TravelAgency的变异算子 48](#_Toc406946752)

[表5-22 TravelAgency故障检测率 49](#_Toc406946753)

[表5-22 TravelAgency故障检测率（续） 50](#_Toc406946754)

[表5-23 TravelAgency实验变异得分 50](#_Toc406946755)

[表5-24 经验研究的算子分布 51](#_Toc406946756)

[表5-25 经验研究的算子检测情况 52](#_Toc406946757)

1. 引言

面向服务的架构(Services-Oriented Architecture, SOA)逐渐成为开发应用程序的主要范型[1]。Web服务是SOA基于Web的一种实现，Web服务是描述一系列操作的接口，它使用标准的、规范的XML描述接口[1]，这一描述中包括与服务进行交互所需要的全部细节，而对外的接口中则隐藏了服务实现的细节，仅仅提供一系列可执行的操作。单一的Web服务无法满足需求，服务组装[2,43]通过将多个Web服务组装起来可以实现复杂、灵活的业务流程。

BPEL [3,4] （Business Process Execution Language）可以将多个相互交互的Web服务编排在一起形成业务流程，并作为一种更为复杂的Web服务对外提供[35]。由于各个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，在实践中，软件测试是一种保证软件可靠性的重要手段 [5,6]。

测试用例的质量决定着软件测试的是否能够高效完成，而变异测试是检测测试用例质量的有效手段[14,15]。变异测试[7,12]是一种基于故障的测试技术，它将着重点放在软件中可能会出现的错误上，具有更强的针对性和高效性。BPEL程序变异测试的有效性很大程度上取决于变异算子的质量[9]。Estero-Botaro等人模拟程序员在实现BPEL2.0组成时可能出现的错误，提出了面向BPEL 2.0的26种变异算子[8]（见附录）。

本文研究面向BPEL程序的变异测试技术与支持工具，对BPEL的26种变异算子进行评估，研究变异算子有效性及其在检测过程中的难易程度，以提高BPEL程序的变异测试效率。课题组提出开发面向BPEL程序的变异体生成系统MuBPEL，该系统根据一组变异算子，自动产生BPEL程序的变异体，完成BPEL的变异测试及结果统计工作。通过多个BPEL程序的经验研究，对BPEL程序故障类型进行总结，得出容易被检测的故障类型和很难被检测的故障类型。BPEL程序的变异测试应该重点检测不易被发现的故障类型，因此精简的变异算子集为BPEL变异测试提供了重要依据，提高了变异测试效率。MuBPEL系统还可以用来评估和比较面向BPEL程序的测试技术的有效性[44,45]。

1. 背景介绍

本章介绍本文相关的概念与技术、国内外研究进展、研究的内容和成果以及论文组织结构。

* 1. 相关概念及技术

简要介绍BPEL以及变异测试的相关知识。

* + 1. BPEL

BPEL[3,4]（Business Process Execution Language，即业务流程执行语言）是一种基于XML的可执行服务组装语言，它将多个独立的Web服务组合成复杂的业务流程，完成更大的业务需求。采用BPEL描述的组合服务能够以Web服务的形式向外提供功能。通过这种方式，BPEL程序可以参与到更高层次的业务流程当中，从而实现服务复用。BPEL有1.0[4]和2.0[3]两个版本，本文以BPEL2.0为研究对象。

BPEL 利用了若干XML 规范：WSDL 1.1，XML Schema 1.0，Xpath 1.0 和XSLT 1.0[3]。BPEL程序由WSDL 消息和XML Schema 类型定义提供数据模型，由Xpath和XSLT提供数据操作支持，并由WSDL描述Web服务的对外接口。

BPEL的基本结构实例如图2-1所示，BPEL程序结构分为四个主要部分[10]：

1. 伙伴链接（<partnerLinks>）：为两个需要进行会话的活动提供了方便，并为每个伙伴提供了“角色”的相关定义。每个<partnerLink>由partnerLinkType 和一个或两个角色名字来标识。
2. 变量（<variables>）：定义程序使用的数据变量，包括三种类型：XML Schema简单类型、XML Schema元素以及WSDL消息类型，变量允许流程在消息交换时维持状态。
3. 故障处理程序（<faultHandlers>）部分包括故障处理器定义了在调用评估和认可服务时对产生的故障必须响应的活动。
4. BPEL程序的活动说明。

上面所提到的所有元素默认情况下具有全局作用域，可以通过scope定义本地作用域，scope可以将业务流程分为几个部分。

<process name="ncname" targetNamespace="uri"

queryLanguage="anyURI"?

expressionLanguage="anyURI"?

suppressJoinFailure="yes|no"?

enableInstanceCompensation="yes|no"?

abstractProcess="yes|no"?>

<partnerLinks> ... </partnerLinks>

<partners>... </partners>

<variables>... </variables>

<correlationSets> ... </correlationSets>

<faultHandlers>... </faultHandlers>

<compensationHandlers> ... </compensationHandlers>

<eventHandlers> ... </eventHandlers>

activity

</process>

图2-1 BPEL基本结构

BPEL程序的基本成分为活动（Activity），分为基本活动和结构性活动[11]。基本活动为不可再分的活动，包括assign、invoke、receive、reply、throw、wait、empty等。结构化活动提供流程执行所需的控制结构，通常包含一个或多个活动，包括sequence、switch、while、flow、pick等。BPEL活动示例如图2-2所示，flow结构性活动里面包含了四个invoke活动。

<flow> <!--**结构性活动-->**

<links>

<link name="checkFlight-To-BookFlight" Attribute />

</links>

<invoke name="checkFlight" . . . > <!--**基本活动-->**

<sources>

<source linkName="checkFlight-To-BookFlight" Attribute />

</sources>

</invoke>

<invoke name="checkHotel" . . . />

<invoke name="bookFlight" . . . >

<targets>

<target linkName="checkFlight-To-BookFlight" />

</targets>

</invoke>

</flow>

图2-2 BPEL活动示例

BPEL程序提供并发与同步的控制结构。图2-2所示的flow启动一组并发活动，并指定他们之间的并发条件，在以上例子中我们可以看到三个web服务调用并行流活动：checkFlight，checkHotel，checkRentCar，还有bookFlight活动，此活动只有在checkFlight活动完成之后才能运行。活动之间的同步是通过link和target建立起来的，只有源活动已完成，目标活动的链接才被会执行。

* + 1. 变异测试

变异测试（也称变异分析）[7]是一种基于故障的软件测试技术，该种测试方法通过将错误植入程序中来对程序进行测试。将软件中可能潜在的故障尽可能早的检测并将其消除，对软件可靠性的提高具有重要意义。人们发现“一般情况下，实施充分的测试也是不可能的，而只能实施相对充分的测试”[12]。即对于一些故障类型我们可以对其进行相对充分的测试，这导致了基于故障的软件测试技术的出现。

变异测试的基本思想是：将变异算子应用于待测程序从而植入错误，得到错误版本的程序集合，这种错误版本的程序被称为变异体。变异算子是对程序设计语言中典型错误的抽象和概括。每个变异体中包含了对原始程序进行的合乎语法的微小改动，从而模拟开发人员编程时所犯的错误。如果一个测试用例使得变异体的执行行为与待测程序不同，这个变异体就称为“被杀死”。

变异测试的开始是解析原始程序，生成变异体，之后是测试人员制定测试用例并将其导入到系统中，此步骤可以是手工完成也可以是自动化实现。变异测试的前提是原始程序的正确性，当利用测试用例测试原始程序，出现错误，则需要对原始程序进行修改，直到得到预期的输出结果。当原始程序的测试输出与预期相符时，可以展开程序的变异测试。如果在同一测试用例输入，当一个变异体与原始程序的输出不符，那么整个变异体被“杀死”，即其错误类型已被发现，在测试中为死亡状态，不需要再对其进行其余的测试步骤。

测试用例执行完之后，存活的变异体可以划分为两类。一类为可杀死的变异体，由于测试用例不充分而不能将其杀死，需要制定更多或更特别的测试用例将其杀死。另一类是等价变异体。在功能上等价于原始程序的变异体即为等价变异体，该变异体的输出一直与原始程序输出相同，因此，根本就不可能测试用例将原始程序和等价变异体鉴别开来。

一个测试用例集合能杀死所有的变异体，那么该测试用例集合相对于这些变异体是充分的，本文中采用两个指标来度量测试用例集合的充分性。

第一个度量指标为变异得分(Mutation Score)[7]，其定义为：

** （2-1）

其中，*P*为被测程序，*T*为测试数据集合，*M*为变异体的数量，*D*为杀死的变异体数量，*E*为等价的变异体数量。变异得分定义了测试用例集合*T*能够杀死非等价变异体的比例，度量了测试用例集合的充分性，其得分越高测试的越有效。等价变异体的存在降低了变异得分，如果不能将其检测出来，将会消耗巨大的测试计算开销。一旦识别剩余的测试就不必进行，避免了资源的浪费。

第二个度量指标是故障检测率(Fault Discovery Rate, FDR)[7],其定义为：

 （2-2）

其中，*m*为变异体，*T*为测试用例集合，*Nf*为集合T中能够杀死*m*变异体的数量，*Nt*为测试用例集合*T*的总数量，*Ni*为非法测试用例数量。故障检测率定义了测试用例集合*T*能够杀死*m*变异体的比例，其检测率越高说明测试用例*T*杀死*m*变异体的几率越大。

近年来，人们普遍认为变异测试具有很强的故障检测能力[13]。与此同时，变异测试被广泛用来评估测试用例集完备性和某种测试技术有效性[14]。变异测试还可以对其他的测试方法进行评估[12]，并借此评估结果可以提升测试方法的能力。

* 1. 国内外研究现状

为了确保BPEL程序的质量，现阶段主要是采用模型验证与测试两种方法，它们体现在工作流开发的不同阶段。

模型验证主要是应用在软件开发的早期，它不对程序代码进行关注，主要是核查程序的逻辑结构，用于排查死锁，检查程序的各路径是否通畅、可达，以减少程序带来的错误。关于模型验证的研究已经有很多，主要有两种：一种是Hinz]等人[19]提出将BPEL转换成Petri网，用Petri网来表示程序的逻辑结构，核实BPEL程序中的逻辑结构是否合理。基于Petri网适合于描述异步的、并发的计算机系统模型。另一种是Fisteus 等人[20]提出将BPEL程序转化为自动机，通过自动机中对各项活动的模拟，查看各活动路径，核实各条路径是否可达，是否会有死锁产生。杨学红等人[24]提出了BPEL 静态缺陷检测方法，采用扩展有限状态机对缺陷进行建模，通过属性状态变迁条件判断缺陷状态，并在所有控制流汇合节点上合并相同属性状态的状态机实例，从而避免了冗余判断的问题。

随着BPEL程序的日益复杂，模型验证已经无法完全保证BPEL的质量，软件测试逐渐成为了检测BPEL错误的关键。

杜吉昌等人[21]提出了使用XCFG(Extend Control Flow Graph)来表达BPEL程序，XCFG 生成BPEL的测试路径，并将测试路径整合、优化，使其生成并发的测试路径，通过各路径生成测试集，进而完成BPEL的测试，这一方法主要是针对BPEL的白盒单元测试。

Yang等人[21,36-39]将BPEL文档中的顺序、分支和循环三种活动转化为控制流图，进而根据控制流图产生基本路径集。Hong等人[40-42]将BPEL程序转换成数据流程图，利用数据流程图产生测试路径集。

Li等人[22]提出通过CFG产生BFG (BPEL Flow Graph)，利用图的遍历方法，形成测试路径，进而生成其测试用例，该方法是对CFG(Control Flow Graph)的扩展。但是由于BFG的构造方法并不具体，因此只能简单地根据映射关系构造BFG图，关于BPEL文件中词法分析并没有很好地解决办法。而且该方法没有考虑到BPEL的异常处理活动等高级属性，BPEL对预期出现异常的有相应的处理办法，这些异常处理使得程序的结构更加复杂，不能简单地按活动出现的顺序来构造，这样就大大的增加了构造程序结构的复杂度。基于上述原因，BFG对于测试BPEL是不完善的。

课题组提出了面向场景的方法[25]，该方法将BPEL程序抽象成一个模型，从模型中按照覆盖准则生成测试场景，实验证明该方法有效。课题组还提出了有效的BPEL的具体故障定位框架[26]，进行故障定位首先发现程序中存在的错误，并找到对应语句，检查这些语句是否存在故障，该框架只关注Web服务的组装，而不关心Web服务内部的情况。

对于BPEL的测试用例生成，Dong等人[29-33]都提出了相关的测试用例生成方法，他们对测试用例的生成方法进行了大量的研究，但是大多数没有研究生成测试用例的质量。

变异测试对于测试用例质量的评估是一种很有效的技术。变异测试是通过由一组变异算子生成的变异体来进行测试的。Estero-Botaro等人模拟程序员在实现BPEL2.0组成时可能出现的错误，提出了面向BPEL 2.0的26种变异算子[8]，运用变异测试技术，并通过遗传算法驱动[23]，通过健壮性功能来生成变异体集合的子集，这种算法能检测到潜在的等价变异。并且因为使用了遗传算法选择出的都是高质量的变异，这种选择独立于变异算子的数目和类型。

* 1. 研究背景与意义

BPEL是广泛认可的Web服务组装的语言，由于Web服务相互协作的动态性，互联网环境的开放性，服务组装的松耦合性，BPEL程序的可靠性不容忽视。本文研究面向BPEL程序的变异测试技术，并开发面向BPEL程序变异测试支持工具MuBPEL。MuBPEL根据变异算子自动化的生成变异体，并对BPEL进行变异测试，统计分析测试结果，提高BPEL程序变异测试效率，MuBPEL同时使用了设计模式[17,18]的方法对变异算子进行管理，避免变异体生成过程中可能出现的冗余。

本文重点研究面向BPEL程序的变异测试技术与支持工具，并对面向BPEL程序变异测试的变异算子的评估，研究变异算子在检测过程中的难易程度。本文提出面向BPEL程序的变异测试框架，并实现了其支持工具MuBPEL，该工具可自动化的生成变异体并对其进行变异测试。BPEL程序变异测试的有效性很大程度上取决于变异算子的质量[9,16]，本文通过经验研究希望找到容易被杀死的变异算子以及不易被杀死的变异算子，对于容易被杀死的变异算子它们很容易被检测，在测试时可以将工作重心放到不易被杀死的变异算子上，通过精简过的变异测试工作将会减少，并为BPEL变异测试提供依据，提高变异测试效率。

本文的研究工作得到中央高校基本科研业务费资助项目“面向SOA的新型软件测试技术与工具”(FRF-SD-12-015A)资助。

* 1. 研究内容及成果

本文对BPEL程序的变异测试进行研究，开发面向BPEL程序变异测试支持工具，对面向BPEL程序的变异算子进行评估，对BPEL变异算子进行精简。主要研究成果如下：

1. 提出了面向BPEL程序变异测试框架，该框架将BPEL程序文件自动生成变异体，通过生成的变异体对BPEL程序进行变异测试。
2. 设计与实现了面向BPEL程序变异测试支持工具MuBPEL，该工具可以自动化的生成变异体，对BPEL进行变异测试，并对变异测试结果进行统计分析。
3. 利用MuBPEL工具对BPEL程序进行变异测试，根据故障检测率和变异得分评估面向BPEL程序变异算子的有效性。
4. 通过经验研究，结合BPEL程序的变异测试结果，寻找面向BPEL程序变异算子被检测到的难易程度，根据分析总结找出容易被检测的变异算子集和不易被检测的变异算子集，找出精简后的变异算子集，以提高BPEL程序变异测试的效率。
   1. 论文组织结构

本论文的组织结构安排如下：

1. 引言；
2. 背景介绍，主要包括相关概念及技术，国内外研究现状，研究内容及成果和研究意义；
3. 深入讨论面向BPEL程序的变异测试问题，主要包括对面向BPEL程序的变异测试的原理，基本方法和测试步骤；
4. 讨论面向BPEL程序的变异测试工具的设计与实现问题，包括需求分析，系统框架，各模块的详细设计，以及系统的实现与演示；
5. 提出变异算子精简技术，利用开发的系统进行大量的经验研究；通过实例测试结果得出变异算子被检测的难易程度，得出精简的变异算子；
6. 工作的总结以及未来的展望。
7. 面向BPEL程序的变异测试

本章首先介绍面向BPEL程序的变异测试框架及其测试过程，然后通过一个示例来演示上述框架及过程。

* 1. 面向BPEL程序的变异测试框架

变异测试将着重点放在软件中可能会出现的错误上，具有更强的针对性和高效性。面向BPEL变异算子的质量决定着变异测试有效性。由Estero-Botaro等人所提出的面向BPEL的26种变异算子，模拟的是程序员用工具书写BPEL可能出现的错误，它用另一种语法结构替换当前的语法结构，并且保证替换后程序的语法是正确的，但并不保持语义的一致。主要考虑的是BPEL程序描述所产生的错误类型，而对Web服务没有进行修改，BPEL程序通过活动的调用完成对Web服务实现。

提出面向BPEL程序的变异测试框架图如图3-1所示。与其他BPEL测试技术不同，面向BPEL程序的变异测试对于调用的Web服务是黑盒测试，对于BPEL程序是白盒测试，在对BPEL进行变异测试时需要BEPL的流程定义（.bpel）文件，BPEL是一种基于XML的语言，在解析BPEL的时候可以使用DOM4J。

面向BPEL程序的变异测试过程中变异体贯穿着变异测试的整个过程，变异测试的前提是变异体自动生成，包括BPEL文件解析、XML文件读写、变异算子管理以及生成变异体四大模块。对BPEL文件（.bpel）进行解析后，将和变异算子进行匹配并生成列表，找到对应的变异算子的处理方法，结合XML文件读写生成相应的变异体。通过输入的测试用例对原始BPEL程序和变异体分别进行测试，统计测试结果，由大量的测试数据得出变异算子检测的难易程度，以得出在对BPEL程序进行变异测试的时候重点需要检测的变异算子，为BPEL程序的变异测试指明方向，提高测试效率。

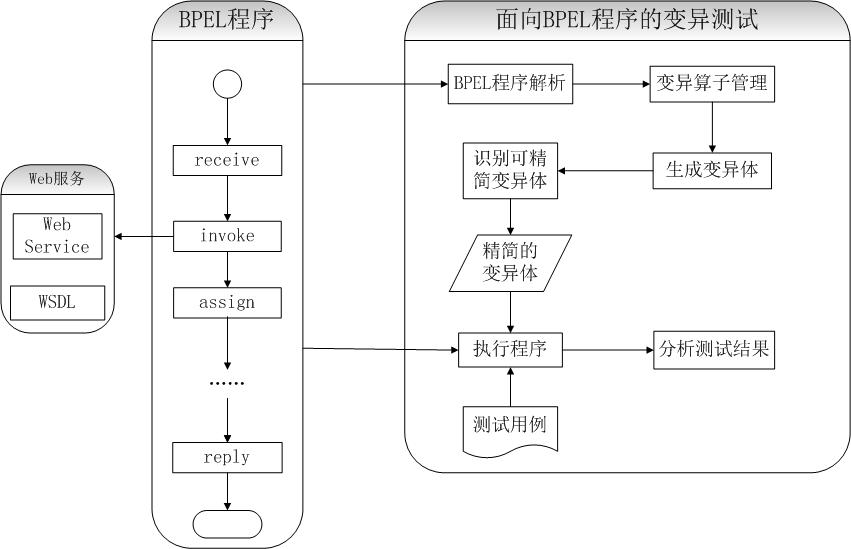


图3-1 面向BPEL程序的变异测试框架图

* 1. 面向BPEL程序的变异测试过程

根据3.1所提出的面向BPEL程序的变异测试框架，需要生成变异体后进行变异测试，通过测试结果来精简变异算子。面向BPEL程序的变异算子变异测试的详细过程如下：

1. 首先，对BPEL程序进行解析。通过解析鉴别可以变异的元素，并与变异算子匹配的代码进行记录。
2. 根据解析记录的变异点找到相关的变异算子管理模块，该模块里存储的是每个变异算子所对应的错误植入规则，26种变异算子的转换规则繁杂且存在着相同类型的转换，可以利用设计模式的方法对变异算子进行管理。
3. 根据变异算子的转换规则，生成对应的变异体。
4. 对于生成的变异体进行识别，得到精简后的变异体
5. 启动BPEL引擎，输入测试用例，执行原始BPEL文件，并记录执行结果。
6. 将生成BPEL程序定义文件的精简变异体重新部署到BPEL环境中，输入上述测试用例，执行变异体文件，与原始BPEL结果进行比较，记录测试结果。
7. 将变异体执行完毕后，统计测试结果，对测试结果进行分析，找出变异算子故障类型被检测的难易程度，对变异算子进行精简，得出精简变异算子。
   1. 示例

本节通过BPELDemo的例子演示面向BPEL程序的变异测试框架及测试过程说明。在BPELDemo存在变异点m，对BPELDemo的变异测试测试步骤为：

1. 上传BPELDemo流程定义文件；
2. 对BPELDemo流程文件进行解析，利用DOM4J将BPELDemo文件作为Document元素返回；
3. 鉴别可以变异的指令或元素，找到变异算子匹配的代码进行记录，获取变异点m；
4. 由变异算子管理模块找到变异点m对应的转换规则；
5. 根据变异算子对m的转换规则对BPELDemo文件进行相应的错误植入操作；
6. 利用DOM4J包将相应的Document修改写入XML文件中，生成变异体M；
7. 对变异体M进行精简，得到精简后的变异体M-；
8. 启动BPEL引擎；
9. 对于给出的BPELDemo文件，分析接口及变量的定义通过测试用例产生方法设置测试用例；
10. 执行原始BPELDemo文件；
11. 对原始程序执行的测试结果进行保存记录；
12. 将变异体M-重新部署到BPEL环境中，利用设置的测试用例变异体文件；
13. 对变异体文件执行的结果和原始程序的结果进行比对，得出测试用例杀死变异体的情况；
14. 统计测试结果，计算故障检测率，产生测试报告；
15. 对测试报告进行分析，评估变异点m被检测的难易程度；
16. 根据分析结果得出变异算子m是否应该被精简。
17. 面向BPEL程序的变异测试系统设计与实现

本章介绍面向BPEL程序的变异测试系统（MuBPEL），MuBPEL支持变异体的自动生成以及对BPEL程序的变异测试，首先分析MuBPEL的需求，然后讨论MuBPEL的系统设计与实现，最后通过一个实例展示MuBPEL。

* 1. 需求分析

BPEL语言将多个Web服务组合在一起以实现更复杂的功能，Web服务具有开放性、松散耦合性及动态性的特点，这些特点给Web服务组装的正确性及准确性带来了挑战，软件测试是确保BPEL质量的重要手段，测试用例的质量决定着软件测试的是否能够高效完成，而变异测试是检测测试用例质量的有效手段。为了更方便高效的对BPEL程序进行变异测试，本文开发了MuBPEL。MuBPEL提供的功能如下：

* + **变异体自动生成：**读取并解析BEPL文件，自动生成变异体。
  + **变异测试执行：**对原始程序和变异体程序进行变异测试。
  + **测试结果分析：**对变异测试结果进行分析总结。

根据功能需求，绘制了面向BPEL程序的变异测试系统对应的用例图，如图3-1所示。

1. 变异体自动生成用例

该用例的主要功能为自动的生成变异体。测试人员提供BPEL待测原始程序，系统读取并对原始程序进行解析，找到变异点及其相关转换规则，生成变异体。包含BPEL程序解析、变异算子管理、变异体生成及XML文件读写四个子用例。

* + BPEL程序解析用例

BPEL变异体的生成需要在BPEL原始程序中植入错误，则要对BPEL程序解析，鉴别可以变异的元素或指令，记录与变异算子匹配的源代码。BPEL是在XML基础之上建立的语言，在解析的时候利用Dom4j解析包，将BPEL程序源文件以Document返回，并对其进行变异点匹配操作。

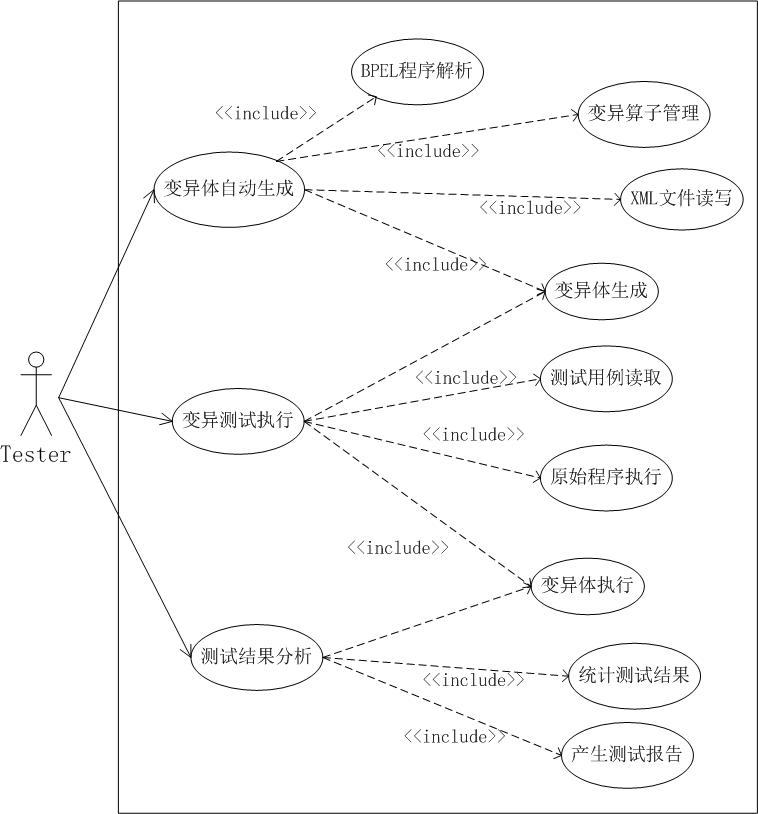


图4-1 MuBPEL的用例图

* + 变异算子管理用例

变异算子管理用例中存储各种转换规则，它通过BPEL解析之后的变异点找到相关的植入错误操作，是变异体生成的关键。该用例需要对26个变异算子进行有效的管理。

* + 变异体生成用例

该用例通过解析得到的与变异算子匹配的原始程序列表，利用算子管理用例中的转换规则，对BPEL原始程序进行相关的转换，将转换后的信息写入BEPL中，该用例同样需要Dom4j技术完成XML写入操作。

* + XML文件读写用例

该用例完成对XML文件的读写功能，借助于JAVA对XML的操作来完成BPEL的读入解析和错误植入，读取BPEL程序利用Dom4j解析返回Document，规则植入时将Document变换写入XML中，完成BPEL的变异体生成。

1. 变异测试执行用例

该用例主要是对BPEL进行变异测试。包括测试用例读取、原始程序执行以及变异体执行三个子用例。

* + 测试用例读取用例

MuBPEL是用来完成BPEL的变异测试的，变异测试不可缺少的就是测试用例，在本系统中，测试用例采用文件导入的方式，测试人员将自己定义的测试用例保存到文件中，系统通过导入的文档读取相关的测试用例。

* + 原始程序执行用例

编写启动BPEL引擎，部署BPEL，利用读取的测试用例对BPEL原始程序进行测试，记录测试结果。

* + 变异体执行用例

将待测的变异体部署到BPEL环境中，利用测试用例对BPEL进行测试，比对测试结果，记录测试状态。

1. 测试结果分析用例

该用例主要是将变异测试结果统计分析。包括统计测试结果和产生测试报告两个子用例。

* + 统计测试结果用例

将变异测试结果进行统计，主要是统计变异体被杀死情况

* + 产生测试报告用例

根据统计结果产生测试报告。

* 1. 系统架构

根据4.1的需求分析，本文提出并设计了面向BPEL程序的变异体自动生成与测试系统MuBPEL，MuBPEL的系统架构图如图4.2所示，由变异体自动生成、变异测试执行及测试结果分析三大模块构成。

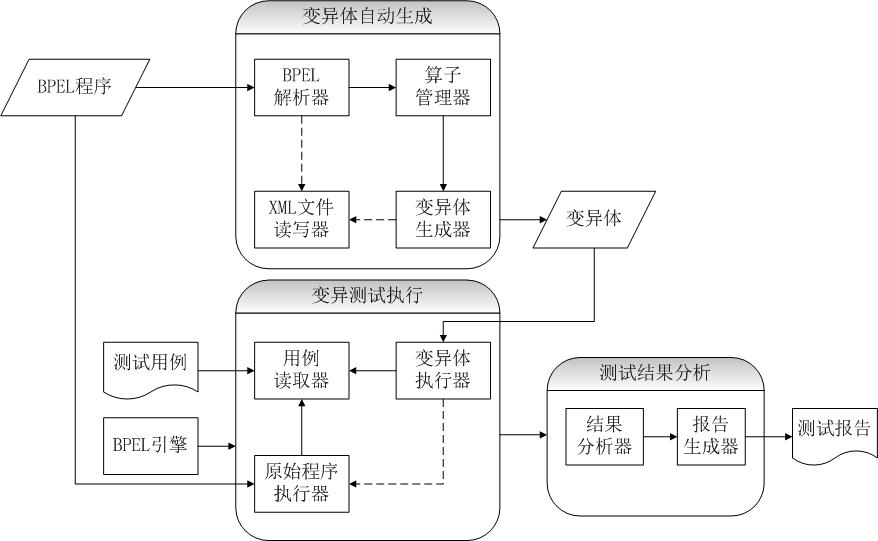


图4-2 MuBPEL的系统架构图

1. 变异体自动生成：该模块的输入是BPEL程序，输出是变异体，包括四个子模块。
   * BPEL解析器：借助XML文件读写器完成对BPEL程序进行解析，找到可变异的元素。
   * 算子管理器：对变异算子的转换规则进行管理。
   * 变异体生成器：通过转换规则，利用XML文件读写器，完成对BEPL流程的错误植入，生成变异体。
   * XML文件读写器：通过对XML文件的读写完成BPEL的读取与写入操作。
2. 变异测试执行：该模块的输入是测试用例、BPEL原始文件以及变异体，输出是变异体的测试结果，在变异测试执行的时候需要启动BPEL引擎，该模块包括三个子模块。

* 用例读取器：将测试人员存储定义的测试用例文件读取。
* 原始程序执行器：利用测试用例对BPEL原始文件进行测试，记录测试结果。
* 变异体执行器：将变异体部署到BPEL环境中进行变异测试，通过比对原始程序的测试结果，得出对于每条测试用例变异体被杀死的情况。

1. 测试结果分析：该模块的输入是变异体测试结果，输出是测试报告，包括两个子模块。

* 结果分析器：统计变异体的故障检测率。
* 报告生成器：生成测试报告，报告的内容为：变异体名称、测试用例个数、杀死变异体测试用例个数、FDR。
  1. 系统设计

根据4.2节的MuBPEL系统架构，设计实现了MuBPEL系统。该系统包括变异体自动生成子系统、变异测试执行子系统、测试结果分析子系统三个模块，本节将会对各模块的设计进行详细的描述。

* + 1. 变异体自动生成子系统设计

BPEL解析器借助XML文件读写器对测试人员上传的BPEL程序文件进行解析，解析后得到可变异的代码与变异算子匹配，变异体生成器根据解析列表调用算子管理器的转换规则，利用XML文件读写器完成变异体的生成过程。图4-3为变异体自动生成时序图。

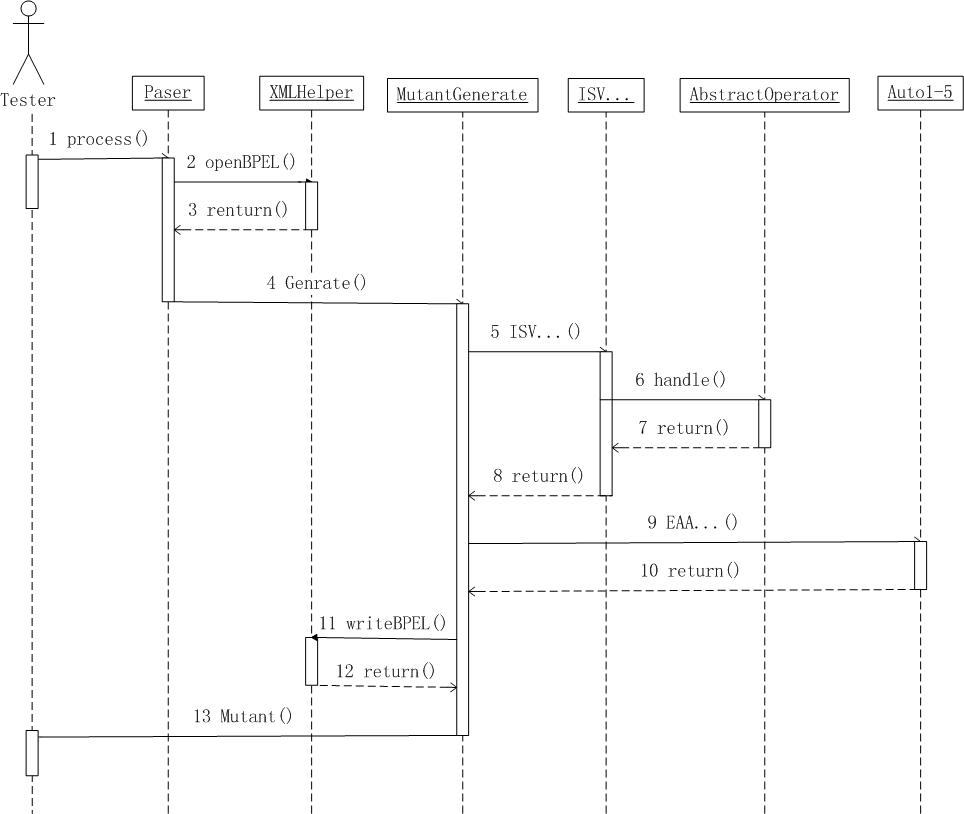


图4-3 变异体自动生成时序图

下面介绍各功能模块。

1. BPEL解析器

BPEL解析器主要是对BPEL程序文件（.bpel）进行解析，鉴别可变异的元素。Parser类负责完成BPEL解析器的任务，主要包括两个方法，分别为process()、getOperatorList()。

Process()方法是对BPEL程序处理的开始，通过调用XML文件读写器的openXMLFile()打开BPEL源文件，找到DOM树根节点，然后从根节点开始遍历，寻找BPEL活动，开始对原始程序与变异算子进行比对，找到可变异的程序。

Process()的解析执行流程为：获取根节点，获得变异元素程序，并判断是否符合各变异算子的转换要求。获取“variable”元素，与ISV变异算子进行比较并记录；获取“if”元素，与AIE、AEL变异算子进行比较并记录；获取“condition”元素，与EAA、 EEU 、ELL 、ERR 、ECC 变异算子进行比较并记录；获取常量数字，与ECN变异算子进行比较并记录；获取“forEach”元素，与AFP变异算子进行比较并记录；获取“while”元素，与AWR变异算子进行比较并记录；获取“receive”元素，与ACI变异算子进行比较并记录；获取“isolated”元素，与AIS变异算子进行比较并记录；获取“joincondition”元素，与AJC变异算子进行比较并记录；获取“pick”元素，与APA、APM变异算子进行比较并记录；获取“sequence”元素，与ASI、ASF变异算子进行比较并记录；获取“wait”、“onAlarm”元素，与EMD变异算子进行比较并记录；获取“until”元素，与EMF变异算子进行比较并记录；获取“eventHandlers”元素，与XEE变异算子进行比较并记录；获取“rethrow”元素，与XER变异算子进行比较并记录；获取“conpensationHandler”元素，与XMC变异算子进行比较并记录；获取“faultHandlers”元素，与XMF变异算子进行比较并记录；获取“terminationHandler”元素，与XMT变异算子进行比较并记录；获取“throw”元素，与XTF变异算子进行比较并记录。

getOperatorList()方法对与变异算子匹配的代码进行记录，最后生成可变异程序的列表List，以供执行模块调用生成变异体。

1. 算子管理器

算子管理器对变异算子进行管理，包Operator里边有26种变异算子的转换规则。包Operator里设计了类继承抽象算子类AbstractOperator.java，并对AbstractOperator.java里的handle()进行重写，在变异体生成时可以对其直接调用，为生成过程提供便利。

26种变异算子转换规则存在着操作相似之处，通过分析比较附录的变异算子，发现部分变异算子非常相似。具体说来：

* EAA、ERR、ELL、ECC变异算子：植入运算符之间替换的错误转换，实现这类变异算子的过程非常相似。
* EMD和EMF变异算子：植入持续时间错误转换，实现方法非常类似，只是“condition”的条件不一样。
* AEL和AIE变异算子：植入活动删除的转换，实现过程相似。
* ACI、AFP、AIS变异算子：植入活动的不同属性改变，如将“yes”属性改为“no”。
* AJC、APM、APA、XMF、XER、XEE、XMC、XMT变异算子：删除某些元素或者删除某个处理器的定义，实现过程相似。

本文在设计与实现算子管理器时，包Operator里包含ISV.java、EEU.java、ECN.java、ASF.java、AWR.java、ASI.java、ATF.java7个类，上述19个类采用工厂模式实现类似的五类算子，不仅实现了高度的代码复用，而且增加了系统的可扩充性。

* 将运算符之间的替换操作封装成StaticFactor1静态工厂，实现EAA、ERR、ELL、ECC变异算子的相同操作，并继承Auto1类完成相关实现。
* 将时间置0或减半操作封装成StaticFactor2静态工厂，实现EMD和EMF变异算子的相同操作，继承Auto2类完成相关实现。
* 将活动的删除操作封装成StaticFactor3静态工厂，完成AEL和AIE变异算子的相同操作，继承Auto3类完成相关的转换规则。
* 将活动属性的修改操作封装成StaticFactor4静态工厂：完成ACI、AFP、AIS变异算子的相同操作，继承Auto4类完成相关的故障植入。
* 将元素或处理器定义删除操作封装成StaticFactor5静态工厂：完成AJC、APM、APA、XMF、XER、XEE、XMC、XMT变异算子的相同操作，继承Auto5类实现各变异算子的错误植入。

当变异体生成器GenerateMutant()调用转换规则时，若调用ISV.java、EEU.java、ECN.java、ASF.java、AWR.java、ASI.java、ATF.java，则会调用抽象类中的handle()将规则进行重写；调用EAA、ERR、ELL、ECC变异算子时，通过StaticFactor1.java继承Auto1.java，并根据各算子条件完成运算符之间替换的转换规则；调用EMD和EMF变异算子时，通过StaticFactor2.java继承Auto2.java，并根据各算子条件完成时间置0或减半操作转换规则；调用AEL和AIE变异算子时，通过StaticFactor3.java继承Auto3.java，并根据各算子条件完成活动的删除转换规则；调用ACI、AFP、AIS变异算子时，通过StaticFactor4.java继承Auto4.java，并根据各算子条件完成活动属性的修改转换规则；调用AJC、APM、APA、XMF、XER、XEE、XMC、XMT变异算子时，通过StaticFactor5.java继承Auto5.java，并根据各算子条件完成元素或处理器定义删除转换规则。

1. 变异体生成器

变异体生成器根据变异体生成列表寻找对应变异算子的转换规则，完成变异体的生成。MutantGenerate类完成变异体生成器的工作，主要包括MutantGenerate ()和IDGenerator()两个方法。

MutantGenerate ()对BPEL解析器Parser.java生成的List列表进行扫描，并调用相应的handle()方法，生成变异体，在变异体生成的时候调用XML文件读写器的writeXMLFile()进行写操作。

IDGenerator()方法对变异体编号，本系统的ID以List列表中匹配顺序定义，这样方便查询变异体所在的位置。

同一个语句中可能存在多个变异点。例如，在表达式“$input.name = 'coca' and $input.amount &lt;$warehouseAmessage.WarehouseAResponse”中，存在两个关系运算符及一个逻辑运算符，可能的变异点有3个。若一个语句中可以包含多个故障，即同时在多个变异点处产生变异，则生成的变异体数目将大大增加。在我们目前的实现中，仅假设每个语句中只存在一个故障。

本系统的变异体生成名称为Mutant+变异算子+ID，如有变异体MutantASF0和MutantACI1，MutantACI1表示为ACI类型的变异体，且该变异体的程序转换位置位于MutantASF0之后。

1. XML文件读写器

XML文件读写器通过对XML的读写操作完成BPEL文件的读取及植入操作。XMLHelper类负责完成XML文件读写器的功能，主要包括openXMLFile()和writeXMLFile()两个方法。

openXMLFile()读取BPEL文件并作为DOM4J中的Document返回，以便其他类使用DOM4J中的函数对生成的DOM树进行操作。

writeXMLFile()将变异算子转换规则写入XML文件中，该方法将系统中读取的Document写入到BPEL文件里。

* + 1. 变异测试执行子系统设计

用例读取器读取测试人员提供的测试用例文件，源代码执行器根据读取的用例对BPEL文件进行测试，并保存测试结果，变异体执行器执行BPEL变异文件并返回测试结果。图4-4为变异测试执行时序图。

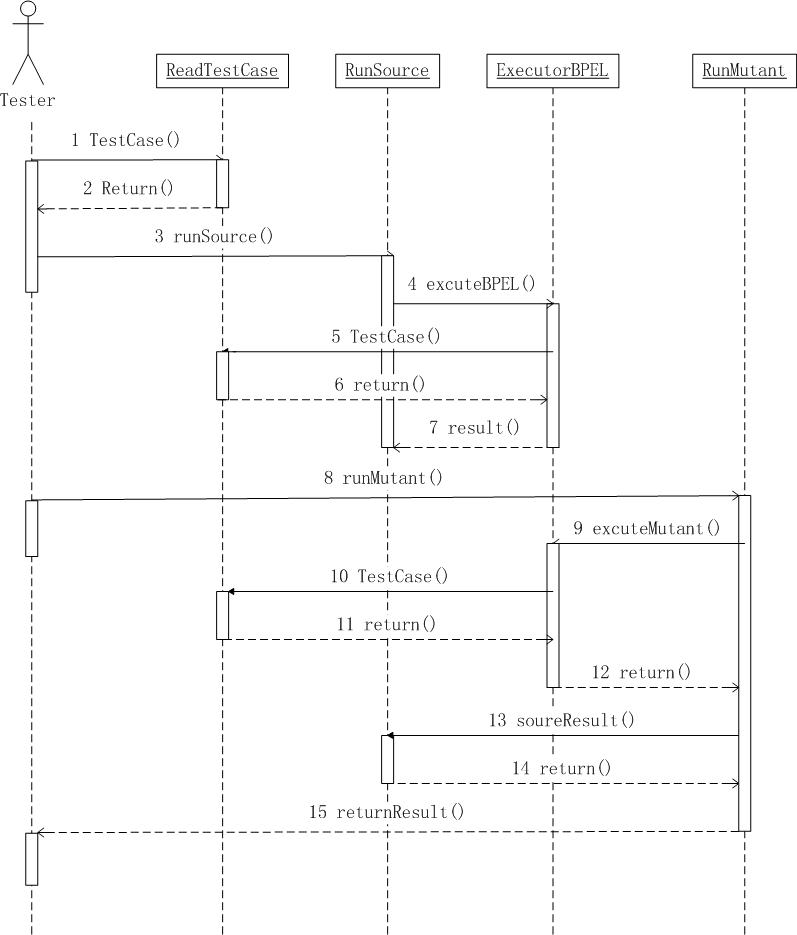


图4-4 变异测试执行时序图

1. 用例读取器

用例读取器读取测试人员导入的测试用例文件并将其显示到系统前端。ReadTestCase类负责完成用例读取器的功能，主要是ReadTestCase ()方法。

用例文档以txt文件上传，一条测试语句的输入以“#”隔开，并以换行符结束，ReadTestCase ()读到空行的时候将会结束获取测试用例，测试用例之间不能出现空行，以防用例不完整读取的情况。

ReadTestCase ()将读取的用例以数组的形式保存，并将输入信息返回到操作界面的No和Input项中（图4-10）。

1. 源代码执行器

源代码执行器根据读取的用例对BPEL源代码进行测试，并将结果保存。RunSource类实现的是对源代码的执行，保存执行结果，在测试之前需要启动BPEL引擎，部署BPEL环境。RunSource.java主要是RunSource()方法，并继承ExecutorBPEL.java。

ExecutorBPEL.java类实现BPEL的执行操作，包括ExecutorBPEL()与TestClient()两种方法。TestClient()方法将ReadTestCase ()读取的测试用例与执行入口进行匹配。ExecutorBPEL()获取“Reply”抛出的执行结果。

RunSource()方法将ExecutorBPEL()获取的结果保存，并将其返回到操作界面的“Source Result”项中（图4-11）。

1. 变异体执行器

变异体执行器完成通过读取的用例对变异体进行测试，根据源代码执行结果返回执行状态。RunMutant类实现的是对变异体的执行，返回测试结果，在对变异体执行之前需要将变异体部署到BPEL环境中。RunMutant.java主要是RunMutant()方法，并继承ExecutorBPEL.java。

RunMutant()获取ExecutorBPEL()执行结果，并将测试结果与“Source Result”各项进行比对，结果相同，则表示该项测试用例不能杀死此变异体，“Mutant Status”显示“unKilled”；否则，显示“Killed”（图4-12）。

* + 1. 测试结果分析子系统设计

结果分析器获取变异测试的结果，并对结果进行总结分析，报告生成器产生相应的报告文档。

1. 结果分析器

结果分析器对变异体执行器的数据进行分析总结。ResultAnalysis类实现了变异测试结果的分析，主要是ResultAnalysis()方法。

ResultAnalysis()获取RunMutant()的测试结果信息，并对变异体的故障检测率等信息进行总结计算。

1. 报告生成器

报告生成器将结果分析器的数据以报告的形式显示，并支持对其下载。ResultReport类实现了测试报告的生成，主要是ResultReport()与DownLoadReport()两个方法。

ResultReport()将ResultAnalysis()的结果显示，主要包括“Mutant Name”、“Number of TestCases”、“Number of TestCases to Kill Mutant”、“Fault Discovery Rate”四项（图4-13）。DownLoadReport()可以将结果以txt形式下载保存。

* 1. 系统实现与演示

根据上述系统架构与子系统设计，本文采用Java语言实现了MuBPEL系统。

* + 1. 系统功能

使用MuBPEL系统可以对BPEL程序文件生成变异体，系统根据生成的变异体对BPEL文件进行变异测试，系统对变异体的测试结果进行分析，形成测试报告。MuBPEL的主要功能如下：

1. BPEL变异体的自动生成：系统将对测试人员输入的BPEL程序进行解析，通过算子管理器找到相应的转换规则，变异体生成器根据转换规则生成变异体。
2. 执行变异测试：系统读取导入的测试用例文件，对BPEL文件进行变异测试。
3. 分析测试结果：系统对变异测试结果分析，并给出变异测试报告。
   * 1. 系统演示

本节利用SupplyChain实例来演示MuBPEL系统，SupplyChain是一个供应链管理的BPEL实例（图5-1），面向BPEL程序的变异测试系统的详细操作如下：

1. 首先，进入系统BPEL选择界面，如图4-5。BPEL原始程序选择界面有“Browse”、“Source BPEL”、“Mutant”三个按钮。点击“Browse”按钮，选择SupplyChain原始程序文件，并可通过“Source BPEL”按钮跳到图4-6，进入SupplyChain源代码查看界面。点击“Mutant”进入跳到图4-7变异算子选择界面。

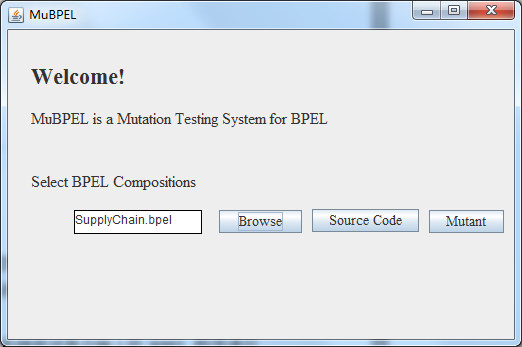


图4-5 BPEL原始程序选择界面

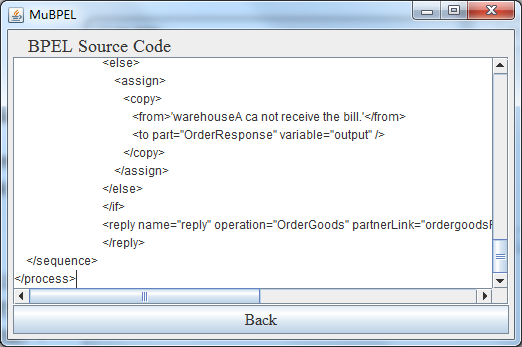


图4-6 BPEL原始程序查看界面

1. 系统根据对原始程序的解析，找到可适用的变异算子，如图4-7。测试人员可在BPEL变异算子选择界面的“Applicable Mutation operators”区域里选择变异算子，鼠标指向变异算子时，界面出现相应的转换规则提示，系统将不适用的算子置灰，测试人员不可操作，只能查看变异算子的描述。实例SupplyChain适用ERR、ELL等8个变异算子，测试人员可对适用的变异算子进行选择。

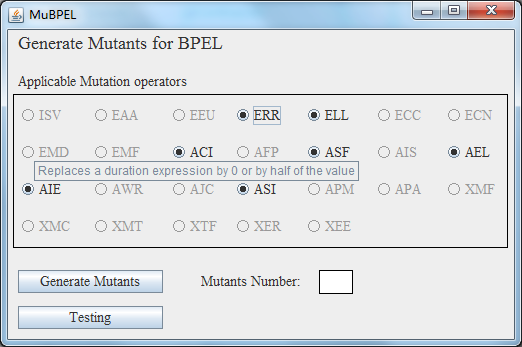


图4-7 BPEL变异算子选择界面

1. 点击“Generate Mutants”按钮，系统将通过测试人员选择的变异算子类型生成变异体，并在界面上显示变异体数量，如图4-8。实例SupplyChain的8个变异算子共产生14个变异体，点击“Testing”按钮，跳到图4-9的MuBPEL变异测试界面。

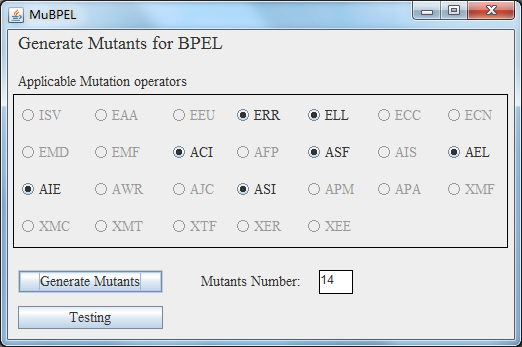


图4-8 BPEL变异体生成界面

1. 图4-9为MuBPEL变异测试界面。在对SupplyChain进行变异测试之前需要先启动BPEL引擎，本例中使用ActiveBPEL引擎测试。点击“Select TestCase”按钮，选择导入测试用例文件。
2. 系统读取用例文件，并将输入显示到MuBPEL的测试用例读取界面，如图4-10。点击“Run Source BPEL”按钮对原始程序进行测试。

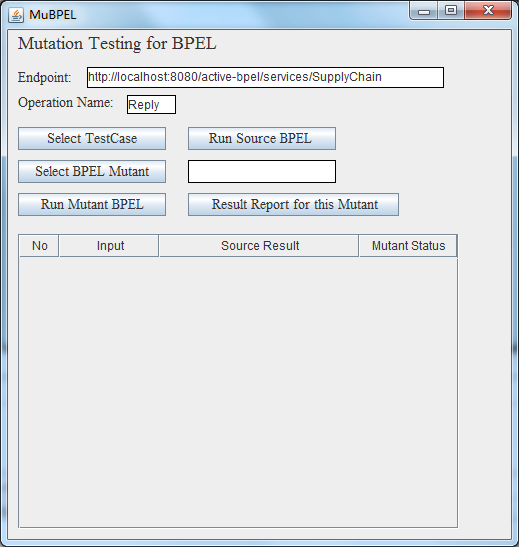


图4-9 MuBPEL变异测试界面

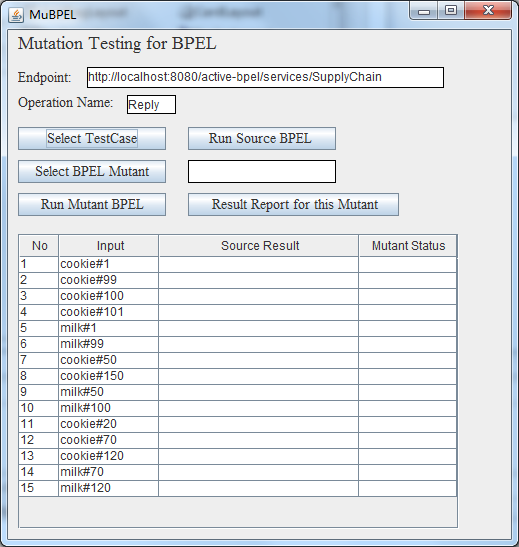


图4-10 MuBPEL测试用例读取界面

1. 系统利用输入对原始程序测试，并将测试结果保存，SupplyChain原始程序的输出如图4-11，MuBPEL原始程序测试界面将“Source Result” 保存并显示。

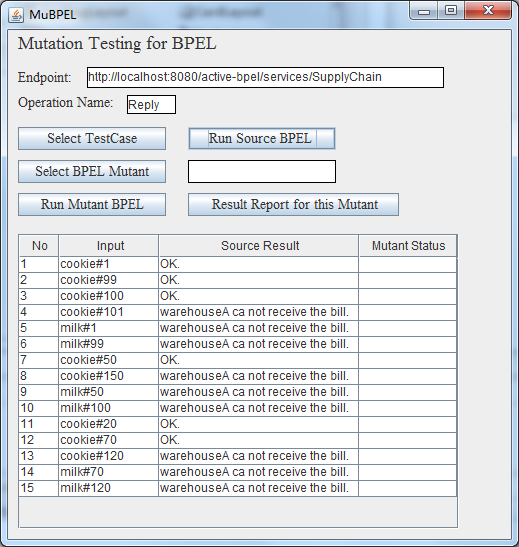


图4-11 MuBPEL原始程序测试界面

1. 点击“Select BPEL Mutant”按钮，选择待测的变异体，并将该变异体部署到BPEL环境上，点击“Run Mutant BPEL”按钮，进行测试，系统将变异体的测试状态显示在“Mutant Status”项里，如图4-12的MuBPEL变异体测试界面。SupplyChain实例的MutantACI1变异体的“Mutant Status”状态全为“Killed”，点击“Result Report for this Mutant”按钮，跳转到图4-12的MuBPEL变异体报告界面。

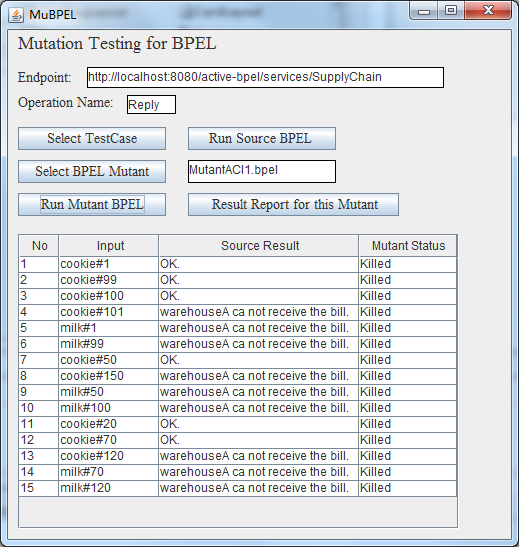


图4-12 MuBPEL变异体测试界面

1. 在变异体报告界面显示变异体测试的结果，包括变异体名称、杀死变异体用例个数、故障检测率（FDR）等信息。点击“Download”按钮，可将测试报告下载；点击“Back”按钮返回图4-11的原始程序测试界面，可接着对SupplyChain的其他变异体进行测试；点击“Exit MuBPEL”按钮，退出系统。

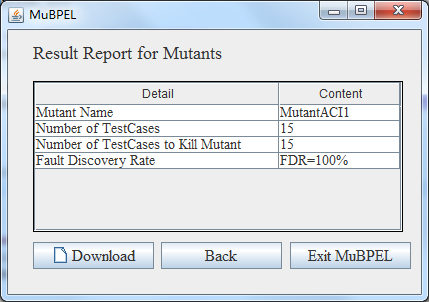


图4-13 MuBPEL变异体报告界面

* 1. 小结

本章介绍了面向BPEL程序的变异测试系统MuBPEL的设计与实现。MuBPEL支持逐个或批量地执行变异测试，借助MuBPEL测试人员可以对BPEL程序进行高效的变异测试。

1. 经验研究

FDR和MS为变异测试的两个重要的度量指标，FDR表示变异体被检测的难易程度，MS表示的是测试用例的充分程度。在本文通过MS和FDR来对变异体进行精简。

变异算子A应用到BPEL程序X上产生变异体集合M，如果给定的测试用例集合T能将变异体集合M全部杀死，那么变异算子A对于X是较易发现的，它的存在对于X的变异测试工作贡献较小，在本文中，力求找到更难检测的变异体。

除了FDR很直观的观测变异算子被检测的难易程度，本文还提出了根据变异算子被检测的相对难易来精简变异算子的方法。变异算子之间的相对精简的思想是：将变异算子提取出来，并分为A、B两组，由A组变异算子生成的变异体集合定为MA，由B组变异算子生成的变异体集合定为MB，对于任意的测试用例集合T，在能杀死MB的基础上，MA都能杀死，则MB较MA更难检测，A的变异算子可以精简。

本章利用已开发的面向BPEL程序的变异测试系统MuBPEL对多个BPEL程序进行变异测试，进一步评估与比较不同类型变异算子的故障检测率。

* 1. 实例SupplyChain的变异测试

使用MuBPEL系统对SupplyChain进行变异测试，并对测试结果进行分析，得出SupplyChain的变异算子精简情况。

1. 待测程序

SupplyChain例子是一个供应链管理的BPEL例子，客户输入需要商品的名称和数量，零售商根据订货单及供应商库存情况向测试人员反馈信息，涉及3个Web服务，图5-1为SupplyChain的执行流程。

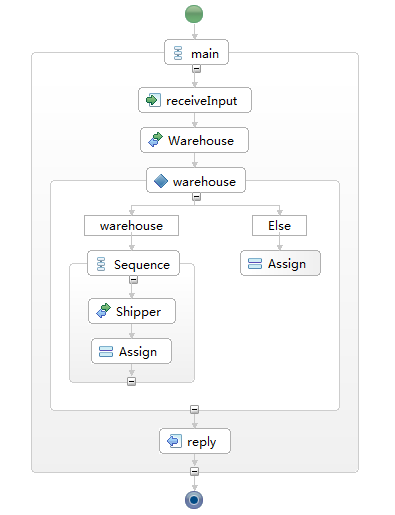


图5-1 SupplyChain执行流程

1. 测试用例设计

对SupplyChain的变量进行分析，其输入输出信息见表5-1所示。

由表5-1可以得出，SupplyChain一共有2条路径，业务流程比较简单，库存的情况如表5-2所示。根据库存情况，通过边界值与等价类划分的方法，设计了TXTYTZ三组测试用例集合，其中TX有6个测试用例，TY有10个测试用例，TZ有15个测试用例。

表5-1 SupplyChain的输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **变量** | **名称** | **类型** | **变量值** | **说明** |
| 输入变量 | name | string | ---- | 商品需求名称 |
| amount | int | ---- | 商品需求数量 |
| 输出变量 | OrderResponse | string | ---- | 供应商库存可供应需求，供应商发出可订货消息 |
| warehouseA ca not receive the bill. | 供应商库存供不应求 |

表5-2 SupplyChain的数据库

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **name** | **Warehouse** | **备注** |
| 1 | cookie | 100 | 可供订货 |
| 2 | Coco | 10 | 不可订货 |
| 3 | milk | 0 | 商品售罄 |

1. 变异体生成

利用设计的MuBPEL对SupplyChain生成变异体，共生成了14个变异体，涉及到的变异算子有7种，除去ERR所占比例较大，其他的算子比较平均。如表5-3所示。

表5-3 SupplyChain的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 7.14% |
| AEL | 2 | 14.29% |
| AIE | 1 | 7.14% |
| ASI | 2 | 14.29% |
| ASF | 2 | 14.29% |
| ERR | 5 | 35.71% |
| ELL | 1 | 7.14% |

1. 测试结果及分析

利用MuBPEL对TXTYTZ三组测试用例集合对SupplyChain原始程序和变异体进行变异测试，根据得到的测试数据，整理结果如表5-4所示。

根据实验的数据，得出以下结论：

1. 对于ACI变异算子，MACI故障检测率为100%，说明此类型的变异体很容易被杀死。
2. 对于ASI和ASF变异算子，MASI与MASF的故障检测率相同。
3. 对于ERR变异算子，MERR故障检测率变化比较大，说明此类型的变异体被测杀的概率不稳定。
4. 对于AEL和AIE变异算子，MAEL中的{MutantAEL12}与MAIE的故障检测率相同，对于测试用例集合TXTYTZ，在能杀死MAEL的基础上MAIE都能杀死，即MAEL包含MAIE，相对于AEL算子AIE可以精简。
5. 对于ELL变异算子，MELL故障检测率变化比较小，说明此类型的变异体被测杀的概率比较稳定。

表5-4 SupplyChain故障检测率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体** | **TX（6）** | **TY（10）** | **TZ（15）** |
| ACI | MutantACI1 | 100% | 100% | 100% |
| AEL | MutantAEL10 | 50% | 40% | 40% |
| MutantAEL12 | 50% | 60% | 60% |
| AIE | MutantAIE11 | 50% | 60% | 60% |
| ASI | MutantASI2 | 100% | 100% | 100% |
| MutantASI9 | 50% | 40% | 40% |
| ASF | MutantASF0 | 100% | 100% | 100% |
| MutantASF8 | 50% | 40% | 40% |
| ERR | MutantERR3 | 33.3% | 30% | 33.3% |
| MutantERR4 | 66.7% | 70% | 66.7% |
| MutantERR5 | 100% | 100% | 100% |
| MutantERR6 | 83.3% | 90% | 93.3% |
| MutantERR7 | 16.7% | 10% | 6.7% |
| ELL | MutantELL13 | 16.7% | 20% | 20% |

用TXTYTZ三组测试用例可以杀死全部的变异体，其变异得分都为100%（如表5-5所示），说明SupplyChain变异测试设计的三组测试用例具有很强的检测能力。

表5-5 SupplyChain实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（6）** | **TY（10）** | **TZ（15）** |
| 变异得分 | 100% | 100% | 100% |

* 1. 实例SmartShelf的变异测试

使用MuBPEL系统对SmartShelf进行变异测试，并对测试结果进行分析，得出SmartShelf的变异算子精简情况。

1. 待测程序

SmartShelf例子是一个货架商品管理的BPEL例子，测试人员输入需要商品的名称和数量，管理系统查询商品信息后，给测试人员反馈相关信息，涉及14个Web服务，业务流程比较复杂，图5-2为SmartShelf的执行流程。

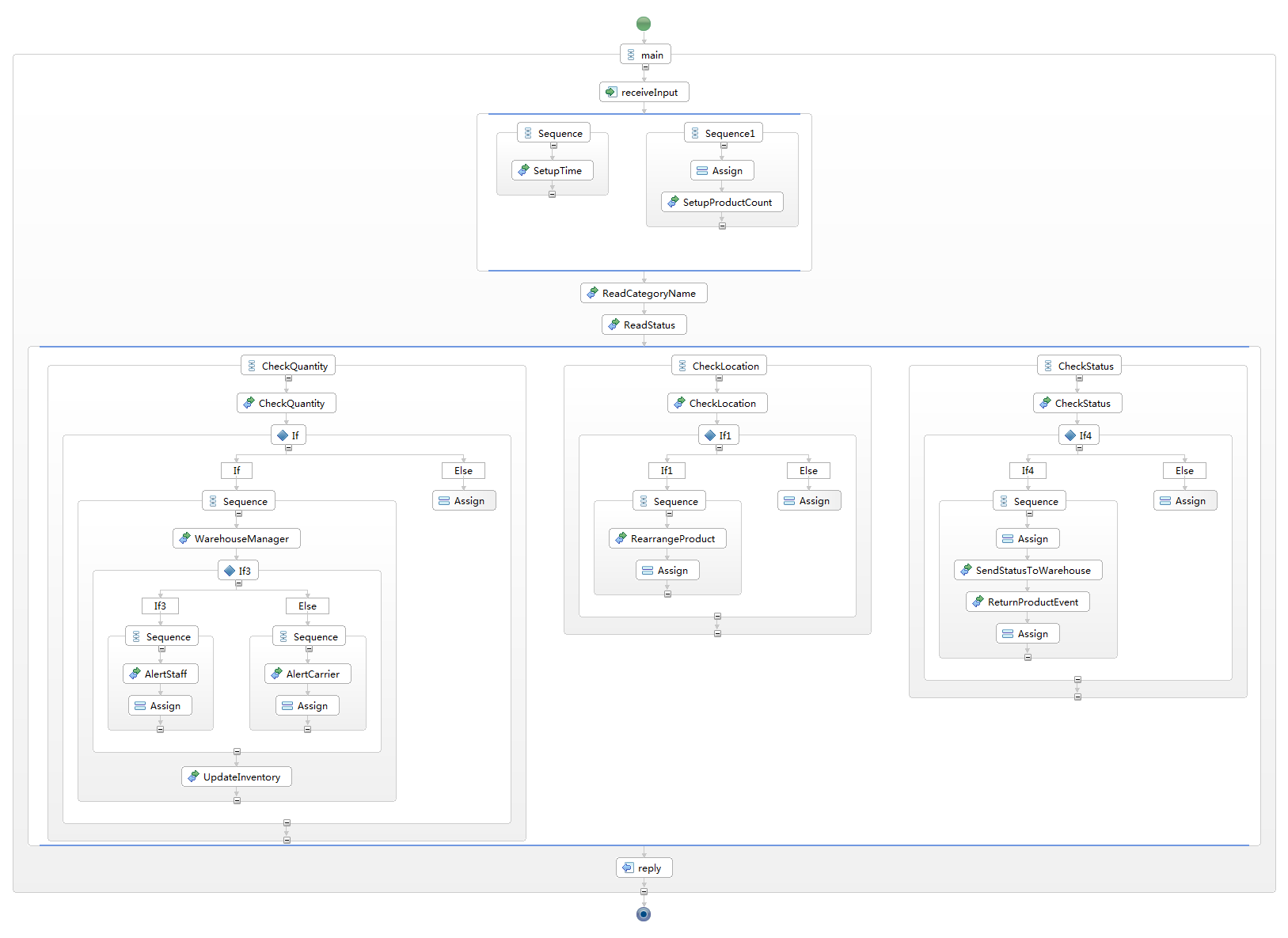


图5-2 SmartShelf执行流程

1. 测试用例设计

SmartShelf货架商品管理的输入输出信息见表5-6所示，测试人员输入需要商品的名称和数量，管理系统查询商品信息后，给测试人员反馈商品的数量、位置和状态三条信息。

由表5-6可以得出，SmartShelf一共有12条路径，为方便测试，我们为商品储存信息设计了一个数据库，具体信息见表5-7。

结合表4-1与表4-2，通过边界值与等价类划分的方法，设计了TXTYTZ三组测试用例集合，其中TX有22个测试用例，TY有35个测试用例，TZ有50个测试用例。

1. 变异体生成

利用设计的MuBPEL对SmartShelf生成变异体，共生成了60个变异体，涉及到的变异算子有8种。其中ERR算子所占比例达到33.3%，而ACI和EEU只有1.7%，其他算子的分布较平均，如表5-8所示。

表5-6 SmartShelf的输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **变量** | **名称** | **类型** | **变量值** | **说明** |
| 输入变量 | name | string | ---- | 商品需求名称 |
| amount | int | ---- | 商品需求数量 |
| 输出变量 | quantity | string | Quantity is enough. | 货架商品数量可供应需求 |
| Warehouse Quantity is enough. | 仓库商品数量可供应需求 |
| Warehouse Quantity is not enough. | 仓库商品数量供不应求 |
| location | string | Location is ok. | 货架商品排放正确 |
| Rearrange is done. | 货架商品排放错误 |
| status | string | Status is ok. | 商品处于正常状态 |
| Status is not overdue. | 商品处于过期状态 |

表5-7 SmartShelf的数据库

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ID** | **name** | **amount** | **Warehouse** | **status** | **location** | **备注** |
| 1 | cookie | 100 | 200 | 2015-05-09 | 1 | 正常商品 |
| 2 | milk | 100 | 0 | 2013-09-09 | 2 | 过期商品 |
| 3 | coca | 0 | 100 | 2015-04-01 | 2 | 商品售罄 |
| 4 | candy | 100 | 300 | 2015-07-15 | 5 | 货架错误 |

表5-8 SmartShelf的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 1.7% |
| AEL | 9 | 15% |
| AIE | 4 | 6.7% |
| ASI | 10 | 16.7% |
| ASF | 11 | 18.3% |
| ERR | 20 | 33.3% |
| EAA | 4 | 6.7% |
| EEU | 1 | 1.7% |

表5-9 SmartShelf故障检测率

| **类型** | **变异体** | **TX（22）** | **TY（35）** | **TZ（50）** |
| --- | --- | --- | --- | --- |
| ACI | MutantACI1 | 100% | 100% | 100% |
| AEL | MutantAEL6 | 100% | 100% | 100% |
| MutantAEL23 | 18.2% | 20% | 28% |
| MutantAEL27 | 40.9% | 37.1% | 36% |
| MutantAEL29 | 40.9% | 40% | 36% |
| MutantAEL39 | 31.8% | 28.6% | 28% |
| MutantAEL41 | 68.2% | 71.4% | 72% |
| MutantAEL51 | 18.2% | 20% | 22% |
| MutantAEL52 | 18.2% | 20% | 22% |
| MutantAEL54 | 81.9% | 80% | 80% |
| AIE | MutantAIE24 | 40.9% | 37.1% | 36% |
| MutantAIE28 | 40.9% | 40% | 36% |
| MutantAIE40 | 68.2% | 71.4% | 72% |
| MutantAIE53 | 81.9% | 80% | 80% |
| ASI | MutantASI2 | 100% | 100% | 100% |
| MutantASI5 | 100% | 100% | 100% |
| MutantASI8 | 100% | 100% | 100% |
| MutantASI15 | 100% | 100% | 100% |
| MutantASI22 | 0 | 0 | 0 |
| MutantASI26 | 0 | 0 | 0 |
| MutantASI31 | 100% | 100% | 100% |
| MutantASI38 | 0 | 0 | 0 |
| MutantASI43 | 100% | 100% | 100% |
| MutantASI50 | 18.2% | 20% | 22% |
| ASF | MutantASF0 | 100% | 100% | 100% |
| MutantASF3 | 0 | 0 | 0 |
| MutantASF4 | 100% | 100% | 100% |
| MutantASF7 | 100% | 100% | 100% |
| MutantASF14 | 100% | 100% | 100% |
| MutantASF21 | 0 | 0 | 0 |
| MutantASF25 | 0 | 0 | 0 |
| MutantASF30 | 100% | 100% | 100% |
| MutantASF37 | 0 | 0 | 0 |
| MutantASF42 | 100% | 100% | 100% |

表5-9 SmartShelf故障检测率（续）

| **类型** | **变异体** | **TX（22）** | **TY（35）** | **TZ（50）** |
| --- | --- | --- | --- | --- |
| ASF | MutantASF49 | 18.2% | 20% | 22% |
| ERR | MutantERR9 | 13.6% | 8.6% | 6% |
| MutantERR10 | 86.4% | 91.4% | 94% |
| MutantERR11 | 100% | 100% | 100% |
| MutantERR12 | 72.7% | 68.6% | 70% |
| MutantERR13 | 59.1% | 60% | 64% |
| MutantERR16 | 13.6% | 8.6% | 6% |
| MutantERR17 | 40.9% | 42.9% | 48% |
| MutantERR18 | 59.1% | 54.3% | 56% |
| MutantERR19 | 40.9% | 31.2% | 28% |
| MutantERR20 | 18.2% | 22.9% | 28% |
| MutantERR32 | 100% | 100% | 100% |
| MutantERR33 | 31.8% | 28.6% | 28% |
| MutantERR34 | 100% | 100% | 100% |
| MutantERR35 | 0 | 0 | 0 |
| MutantERR36 | 68.2% | 71.4% | 72% |
| MutantERR44 | 0 | 0 | 0 |
| MutantERR45 | 100% | 100% | 100% |
| MutantERR46 | 100% | 100% | 100% |
| MutantERR47 | 18.2% | 20% | 22% |
| MutantERR48 | 81.9% | 80% | 80% |
| EAA | MutantEAA55 | 63.6% | 62.9% | 68% |
| MutantEAA56 | 63.6% | 62.9% | 68% |
| MutantEAA57 | 63.6% | 62.9% | 68% |
| MutantEAA58 | 63.6% | 62.9% | 68% |
| EEU | MutantEEU59 | 63.6% | 62.9% | 68% |

1. 测试结果及分析

利用6.2.2的TXTYTZ三组测试用例集合对SmartShelf原始程序和变异体进行变异测试，根据得到的测试结果，整理数据如表5-9所示。

根据实验的数据，得出以下结论：

1. 对于ACI变异算子，MACI故障检测率为100%，说明此类型的变异体很容易被杀死。
2. 对于AEL和AIE变异算子，MAEL中的{MutantAEL27, MutantAEL 29, MutantAEL 41, MutantAEL 54}与MAIE的故障检测率相同，对于测试用例集合TXTYTZ，在能杀死MAEL的基础上MAIE都能杀死，即MAEL包含MAIE，相对于AEL算子AIE可以精简。
3. 对于ASI和ASF变异算子，MASI中的{MutantASI22, MutantASI26, MutantASI 38}与MASF的{MutantASF3, MutantASF 21, MutantASF 25, MutantASF 27}为等价变异体，其他的故障检测率相同，相对于ASI算子ASF可以精简。
4. 对于ERR变异算子，MERR故障检测率变化比较大，说明此类型的变异体被测杀的概率不稳定。
5. 对于EAA算子，MEAA故障检测率变化比较小，本例原型算术表达式为两变量之差与0的比较，故MEAA与MEEU的故障检测率都相同。

在这60个变异体中，有9个变异体的故障检测率为0，其中有7个等价变异体，分析测试结果数据，可以发现三组测试用例集的变异分数一样（如表5-10），三组测试用例的MS都为96.2%，有两个变异体没有被检测出来，设计的测试用例相对比较充分。

表5-10 SmartShelf实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（22）** | **TY（35）** | **TZ（50）** |
| 变异得分 | 96.2% | 96.2% | 96.2% |

* 1. 实例SupplyCustomer的变异测试

使用MuBPEL系统对SupplyCustomer进行变异测试，并对测试结果进行分析，得出SupplyCustomer的变异算子精简情况。

1. 待测程序

SupplyCustomer例子是一个项目订单管理的例子，客户输入订单的信息和地址，系统通过验证之后向客户反馈信息，涉及6个Web服务，图5-3为SupplyCustomer的执行流程。

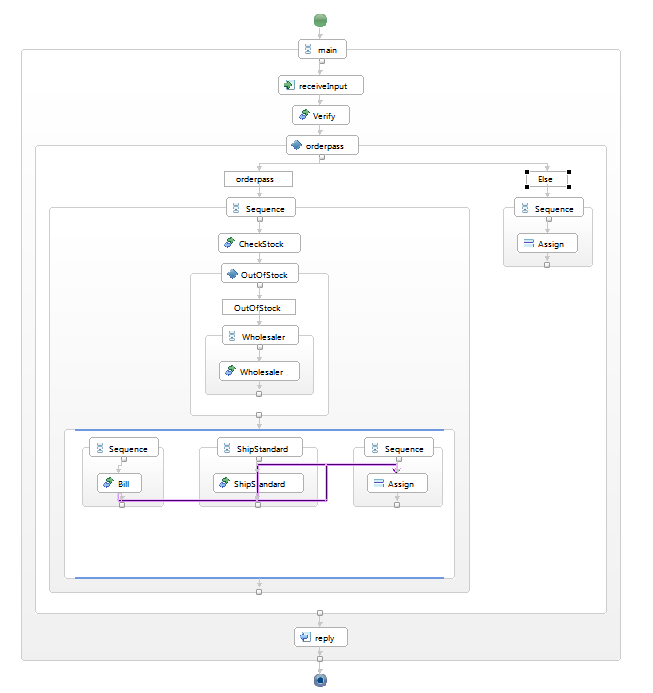


图5-3 SupplyCustomer执行流程

1. 测试用例设计

项目订单管理的输入输出信息见表5-11所示，客户输入订单的信息和地址，系统通过验证之后向客户反馈信息。

表5-11 SupplyCustomer的输入输出

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **变量** | **名称** | **类型** | **变量值** | **说明** |
| 输入变量 | items | string | ---- | 项目订货信息 |
| address | string | ---- | 项目订货地址 |
| 输出变量 | SendOrderResponse | string | yes | 订单检查通过 |
| Order Checked Failure | 订单检查失败 |

SupplyCustomer一共有3条路径，业务流程比较简单，采用表5-11的库存，根据库存情况，通过边界值与等价类划分的方法，设计了TXTYTZ三组测试用例集合，其中TX有7个测试用例，TY有12个测试用例，TZ有18个测试用例。

1. 变异体生成

利用设计的MuBPEL对SupplyCustomer生成变异体，共生成了25个变异体，涉及到的变异算子有8种。其中ERR与ASF所占比例较大，两者之和将近70%，其他6个算子的所占比例则较小，基本平均分布，如表5-12所示。

表5-12 SupplyCustomer的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 4% |
| AEL | 2 | 8% |
| AIE | 1 | 4% |
| ASI | 2 | 8% |
| ASF | 7 | 28% |
| ERR | 10 | 40% |
| ELL | 1 | 8% |
| AJC | 1 | 8% |

1. 测试结果及分析

利用TXTYTZ三组测试用例集合对SupplyCustomer原始程序和变异体进行变异测试，根据得到的数据，计算出测试用例集合的故障检测率，其结果如表5-13所示。

根据实验的数据，得出以下结论：

1. 对于ACI变异算子，MACI故障检测率为100%，说明此类型的变异体很容易被杀死。
2. 对于AEL和AIE变异算子，MAEL中的{MutantAEL22}与MAIE的故障检测率相同，对于测试用例集合TXTYTZ，在能杀死MAEL的基础上MAIE都能杀死，即MAEL包含MAIE，相对于AEL算子AIE可以精简。
3. MASF的{MutantASF15, MutantASF16, MutantASF17, MutantASF18, MutantASF21}为等价变异体
4. MASF的{MutantASF0, MutantASF8}与MASI的故障检测率相同，相对于ASI算子ASF可以精简。
5. 对于ERR变异算子，MERR故障检测率变化比较大，说明此类型的变异体被测杀的概率不稳定。
6. 对于ELL和AJC变异算子，故障检测率为0 ，说明此类型的变异体不易被检测。

表5-13 SupplyCustomer故障检测率

| **类型** | **变异体** | **TX（7）** | **TY（12）** | **TZ（18）** |
| --- | --- | --- | --- | --- |
| ACI | MutantACI1 | 100% | 100% | 100% |
| AEL | MutantAEL19 | 42.9% | 33.3% | 33.3% |
| MutantAEL22 | 57.1% | 66.7% | 66.7% |
| AIE | MutantAIE20 | 57.1% | 66.7% | 66.7% |
| ASI | MutantASI2 | 100% | 100% | 100% |
| MutantASI9 | 85.7% | 83.3% | 83.3% |
| ASF | MutantASF0 | 100% | 100% | 100% |
| MutantASF8 | 85.7% | 83.3% | 83.3% |
| MutantASF15 | 0 | 0 | 0 |
| MutantASF16 | 0 | 0 | 0 |
| MutantASF17 | 0 | 0 | 0 |
| MutantASF18 | 0 | 0 | 0 |
| MutantASF21 | 0 | 0 | 0 |
| ERR | MutantERR3 | 42.9% | 33.3% | 33.3% |
| MutantERR4 | 42.9% | 33.3% | 33.3% |
| MutantERR5 | 0 | 0 | 0 |
| MutantERR6 | 42.9% | 33.3% | 33.3% |
| MutantERR7 | 0 | 0 | 0 |
| MutantERR10 | 85.7% | 83.3% | 83.3% |
| MutantERR11 | 85.7% | 83.3% | 83.3% |
| MutantERR12 | 42.9% | 50% | 50% |
| MutantERR13 | 42.9% | 33.3% | 33.3% |
| MutantERR14 | 0 | 0 | 0 |
| ELL | MutantELL23 | 0 | 0 | 0 |
| AJC | MutantAJC24 | 0 | 0 | 0 |

在这25个变异体中，有10个变异体的故障检测率为0，其中有5个等价变异体，分析测试结果数据，可以发现三组测试用例集的变异分数都为75%（如表5-14），设计的测试用例不够充分，SupplyCustomer实例里的存活的变异体类型不容易被检测。

表5-14 SupplyCustomer实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（7）** | **TY（12）** | **TZ（18）** |
| 变异得分 | 75% | 75% | 75% |

* 1. 实例loan\_approval的变异测试

使用MuBPEL系统对loan\_approval进行变异测试，并对测试结果进行分析，得出loan\_approval的变异算子精简情况。

1. 待测程序

loan\_approval例子是一个贷款审批的例子，涉及6个Web服务，客户输入自己的个人信息和贷款数目，贷款审批执行审批流程后，返回贷款成功或贷款失败。图5-4为loan\_approval的执行流程。

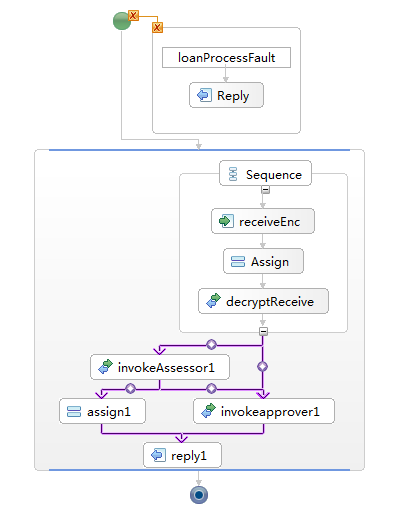


图5-4 loan\_approval执行流程

1. 测试用例设计

loan\_approval一共有3条路径，通过银行人员信息信息情况，设计了TXTYTZ三组测试用例集合，其中TX有12个测试用例，TY有18个测试用例，TZ有25个测试用例。

1. 变异体生成

利用MuBPEL对loan\_approval生成变异体，共生成了24个变异体，涉及到的变异算子有7种。其中ERR与ECN所占比例高达45%，如表5-15所示。

表5-15 loan\_approval的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 4.2% |
| AEL | 2 | 8.3% |
| ASI | 1 | 4.2% |
| ASF | 1 | 4.2% |
| ECN | 8 | 33.3% |
| ERR | 10 | 41.7% |
| XMF | 1 | 4.2% |

1. 测试结果及分析

利用TXTYTZ三组测试用例集合对loan\_approval进行变异测试，根据得到的数据，计算出测试用例集合的故障检测率，其结果如表5-16所示。

根据实验的数据，得出以下结论：

1. 对于ACI、 XMF、AEL变异算子，故障检测率有一定的变化,说明此类型的变异体被测杀的概率不稳定。
2. MASF与MASI的故障检测率相同。
3. 对于ERR和ECN变异算子，他们的故障检测率变化比较大，MECN的故障检测率小于MERR的故障检测率，测试过程中发现：能够杀死MECN的测试用例都能杀死MERR，因此，在带有常量的关系表达式中，ERR相对于ECN是可以精简的。

用TXTYTZ三组测试用例可以杀死全部的变异体，其变异得分如表5-17所示，三组测试用例的MS均为100%，说明设计的测试用例具有很强的检测能力。

表5-16 loan\_approval故障检测率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体** | **TX（12）** | **TY（18）** | **TZ（25）** |
| ACI | MutantACI2 | 66.7% | 66.7% | 68% |
| AEL | MutantAEL4 | 66.7% | 66.7% | 68% |
| MutantAEL23 | 58.3% | 55.6% | 52% |
| ASI | MutantASI3 | 66.7% | 66.7% | 68% |
| ASF | MutantASF1 | 66.7% | 66.7% | 68% |
| ECN | MutantECN10 | 8.3% | 5.6% | 4% |
| MutantECN11 | 16.7% | 11.1% | 8% |
| MutantECN12 | 8.3% | 11.1% | 12% |
| MutantECN13 | 33.3% | 33.3% | 32% |
| MutantECN19 | 16.7% | 11.1% | 8% |
| MutantECN20 | 8.3% | 5.6% | 4% |
| MutantECN21 | 16.7% | 22.2% | 20% |
| MutantECN22 | 8.3% | 11.1% | 12% |
| ERR | MutantERR5 | 8.3% | 5.6% | 4% |
| MutantERR6 | 58.3% | 55.6% | 56% |
| MutantERR7 | 41.7% | 44.4% | 48% |
| MutantERR8 | 33.3% | 33.3% | 32% |
| MutantERR9 | 25% | 22.2% | 24% |
| MutantERR14 | 8.3% | 5.6% | 4% |
| MutantERR15 | 33.3% | 38.9% | 44% |
| MutantERR16 | 41.7% | 44.4% | 48% |
| MutantERR17 | 25% | 27.8% | 32% |
| MutantERR18 | 16.7% | 16.7% | 16% |
| XMF | MutantXMF0 | 33.3% | 33.3% | 32% |

表5-17 loan\_approval实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（12）** | **TY（18）** | **TZ（25）** |
| 变异得分 | 100% | 100% | 100% |

* 1. 实例CarEstimate的变异测试

使用MuBPEL系统对CarEstimate进行变异测试，并对测试结果进行分析，得出CarEstimate的变异算子精简情况。

1. 待测程序

CarEstimate例子是一个汽车修复评估系统，涉及8个Web服务，顾客提供评估请求，系统首先进行初步评估，然后根据请求对汽车进行简单评估或复杂评估，最后调用最终评估将评估结果返回给顾客。图5-5为CarEstimate的执行流程。

1. 测试用例设计

CarEstimate一共有3条路径，设计了TXTYTZ三组测试用例集合，其中TX有10个测试用例，TY有20个测试用例，TZ有30个测试用例。

1. 变异体生成

利用设计的MuBPEL对CarEstimate生成变异体，共生成了21个变异体，涉及到的变异算子有8种，算子之间的分布较平均，如表5-18所示。

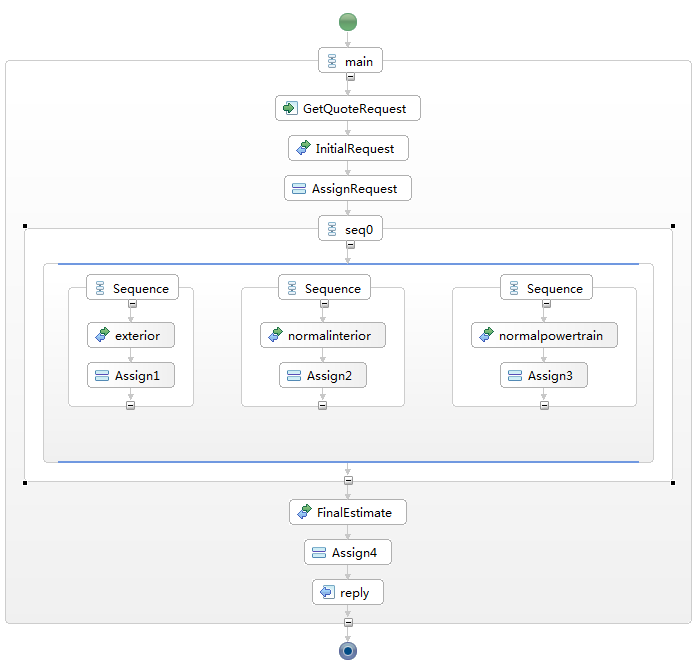


图5-5 CarEstimate执行流程

表5-18 CarEstimate的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 4.8% |
| AEL | 5 | 23.8% |
| ASI | 4 | 14% |
| ASF | 5 | 23.8% |
| EMD | 2 | 9.5% |
| EMF | 2 | 9.5% |
| APM | 1 | 4.8% |
| APA | 1 | 4.8% |

1. 测试结果及分析

利用TXTYTZ三组测试用例集合对CarEstimate进行变异测试，根据得到的数据，计算出测试用例集合的故障检测率，其结果如表5-19所示。

根据实验的数据，得出以下结论：

1. 对于ACI变异算子，故障检测率为100%，说明此类型的变异体很容易被杀死。
2. MAEL和MAPA的故障检测率有一定的变化，说明此类型的变异体被测杀的概率不稳定。
3. 对于ASI和ASF变异算子，MASF中的{MutantASF4}为等价变异体，其他的故障检测率相同，相对于ASI算子ASF可以精简。
4. 对于EMD和EMF变异算子，{MutantEMD16}的故障检测率小于{MutantEMF18}，且能杀死MEMD的测试用例都能杀死MEMF，故EMF相对于EMD是可以精简的。
5. 对于APM变异算子，其故障检测率为100%。

在21个变异体中有一个为等价变异体，其他变异体用TXTYTZ三组测试用例都可以杀死，其变异得分如表5-20所示，设计的三组测试用例具有很强的检测能力。

表5-19 CarEstimate故障检测率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体** | **TX（10）** | **TY（20）** | **TZ（30）** |
| ACI | MutantACI2 | 100% | 100% | 100% |
| AEL | MutantAEL3 | 100% | 100% | 100% |
| MutantAEL7 | 20% | 20% | 20% |
| MutantAEL10 | 40% | 45% | 33.3% |
| MutantAEL13 | 40% | 35% | 46.7% |
| MutantAEL14 | 100% | 100% | 100% |
| ASI | MutantASI1 | 100% | 100% | 100% |
| MutantASI6 | 20% | 20% | 20% |
| MutantASI9 | 40% | 45% | 33.3% |
| MutantASI12 | 40% | 35% | 46.7% |
| ASF | MutantASF0 | 100% | 100% | 100% |
| MutantASF4 | 0 | 0 | 0 |
| MutantASF5 | 20% | 20% | 20% |
| MutantASF8 | 40% | 45% | 33.3% |
| MutantASF11 | 40% | 35% | 46.7% |
| EMD | MutantEMD15 | 0 | 0 | 0 |
| MutantEMD16 | 40% | 45% | 33.3% |
| EMF | MutantEMF17 | 100% | 100% | 100% |
| MutantEMF18 | 60% | 65% | 53.3% |
| APM | MutantAPM19 | 100% | 100% | 100% |
| APA | MutantAPA20 | 40% | 35% | 46.7% |

表5-20 CarEstimate实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（10）** | **TY（20）** | **TZ（30）** |
| 变异得分 | 100% | 100% | 100% |

* 1. 实例TravelAgency的变异测试

使用MuBPEL系统对TravelAgency进行变异测试，并对测试结果进行分析，得出TravelAgency的变异算子精简情况。

1. 待测程序

TravelAgency例子是一个旅行社预订系统，涉及9个Web服务，该系统由旅行社预订、旅店预订、飞机预订和银行结算组成。客户提供个人信息和人数，系统根据人数来选择预订服务，最后将预订信息与结算账单返回非客户。图5-6为TravelAgency的执行流程。

1. 测试用例设计

由图5-5可看出TravelAgency有12条路径，由于旅行社预订系统中个服务之间的绑定，TravelAgency的路径只有2条，设计了TXTYTZ三组测试用例集合，其中TX有4个测试用例，TY有7个测试用例，TZ有10个测试用例。

1. 变异体生成

利用设计的MuBPEL对TravelAgency生成变异体，共生成了50个变异体，涉及到的变异算子有8种。其中ERR算子所占比例高达40%，ACI与XMC所占比例为2%，其他5个算子平均分布，如表5-21所示。

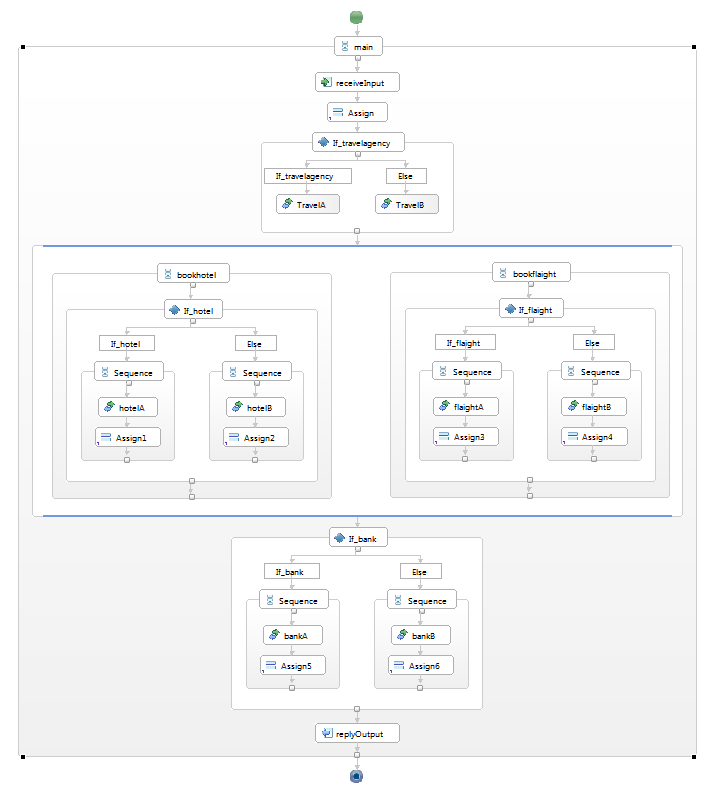


图5-6 TravelAgency执行流程

表5-21 TravelAgency的变异算子

|  |  |  |
| --- | --- | --- |
| **类型** | **个数** | **所占百分比** |
| ACI | 1 | 2% |
| AEL | 7 | 14% |
| AIE | 5 | 10% |
| ASI | 4 | 8% |
| ASF | 6 | 12% |
| ERR | 20 | 40% |
| ISV | 6 | 12% |
| XMC | 1 | 2% |

1. 测试结果及分析

利用TXTYTZ三组测试用例集合对TravelAgency进行变异测试，根据得到的数据，计算出测试用例集合的故障检测率，其结果如表5-22所示。

根据测试结果及实验的数据，得出以下结论：

1. 对于ACI变异算子，MACI故障检测率为100%，说明此类型的变异体很容易被杀死。
2. 对于AEL和AIE变异算子，MAEL中的{MutantAEL9, MutantAEL 17, MutantAEL 21, MutantAEL 32，MutantAEL 42}与MAIE的故障检测率相同，对于测试用例集合TXTYTZ，在能杀死MAEL的基础上MAIE都能杀死，即MAEL包含MAIE，相对于AEL算子AIE可以精简。
3. 对于ASI和ASF变异算子， MASF的{MutantASF10, MutantASF22}为等价变异体，其他的故障检测率相同，相对于ASI算子ASF可以精简。
4. 对于ERR变异算子，MERR故障检测率变化比较大，说明此类型的变异体被测杀的概率不稳定。
5. 对于ISV变异算子，该例中由于各变量之间的互相绑定，此类型的变异体被测杀的故障检测率比较稳定。
6. 对于XMC变异算子，该类的变异体不易被杀死。

在50个变异体中有3个故障检测率为0，其中2个为等价变异体，其他变异体用TXTYTZ三组测试用例都可以杀死，其变异得分如表5-23所示，三组测试用例的变异得分均为96%，说明设计的测试用例相对充分。

表5-22 TravelAgency故障检测率

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体** | **TX（4）** | **TY（7）** | **TZ（10）** |
| ACI | MutantACI2 | 100% | 100% | 100% |
| AEL | MutantAEL9 | 50% | 57.1% | 50% |
| MutantAEL17 | 50% | 42.9% | 50% |
| MutantAEL21 | 50% | 57.1% | 50% |
| MutantAEL28 | 50% | 42.9% | 50% |
| MutantAEL32 | 50% | 57.1% | 50% |
| MutantAEL38 | 50% | 42.9% | 50% |
| MutantAEL42 | 50% | 57.1% | 50% |
| AIE | MutantAIE8 | 50% | 57.1% | 50% |
| MutantAIE16 | 50% | 42.9% | 50% |
| MutantAIE20 | 50% | 57.1% | 50% |
| MutantAIE31 | 50% | 57.1% | 50% |
| MutantAIE41 | 50% | 57.1% | 50% |
| ASI | MutantASI1 | 100% | 100% | 100% |
| MutantASI19 | 50% | 42.9% | 50% |
| MutantASI30 | 50% | 42.9% | 50% |
| MutantASI40 | 50% | 42.9% | 50% |
| ASF | MutantASF0 | 100% | 100% | 100% |
| MutantASF10 | 0 | 0 | 0 |
| MutantASF18 | 50% | 42.9% | 50% |
| MutantASF22 | 0 | 0 | 0 |
| MutantASF29 | 50% | 42.9% | 50% |
| MutantASF39 | 50% | 42.9% | 50% |
| ERR | MutantERR3 | 25% | 14.3% | 10% |
| MutantERR4 | 75% | 57.1% | 60% |
| MutantERR5 | 25% | 42.9% | 40% |
| MutantERR6 | 25% | 85.7% | 90% |
| MutantERR7 | 100% | 100% | 100% |
| MutantERR11 | 25% | 14.3% | 10% |
| MutantERR12 | 75% | 57.1% | 60% |
| MutantERR13 | 25% | 42.9% | 40% |
| MutantERR14 | 25% | 85.7% | 90% |

表5-22 TravelAgency故障检测率（续）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体** | **TX（4）** | **TY（7）** | **TZ（10）** |
| ERR | MutantERR15 | 100% | 100% | 100% |
| MutantERR23 | 25% | 14.3% | 10% |
| MutantERR24 | 75% | 57.1% | 60% |
| MutantERR25 | 25% | 42.9% | 40% |
| MutantERR26 | 25% | 85.7% | 90% |
| MutantERR27 | 100% | 100% | 100% |
| MutantERR33 | 25% | 14.3% | 10% |
| MutantERR34 | 75% | 57.1% | 60% |
| MutantERR35 | 25% | 42.9% | 40% |
| MutantERR36 | 25% | 85.7% | 90% |
| MutantERR37 | 100% | 100% | 100% |
| ISV | MutantISV43 | 50% | 57.1% | 50% |
| MutantISV44 | 50% | 42.9% | 50% |
| MutantISV45 | 50% | 57.1% | 50% |
| MutantISV46 | 50% | 42.9% | 50% |
| MutantISV47 | 50% | 57.1% | 50% |
| MutantISV48 | 50% | 42.9% | 50% |
| XMC | MutantXMC49 | 0 | 0 | 0 |

表5-23 TravelAgency实验变异得分

|  |  |  |  |
| --- | --- | --- | --- |
|  | **TX（10）** | **TY（20）** | **TZ（30）** |
| 变异得分 | 96% | 96% | 96% |

* 1. 变异体精简

利用MuBPEL对6个BPEL实例进行变异测试，表5-24总结了变异算子的分布情况。对于26种变异算子，不是每个算子都可以适用到BPEL程序里，从表5-24可以看出，ECC、AFP、AIS、AWR、XMT、XTF、XER、XEE变异算子没有被6个实例所应用，它们只适用于非常特别的BPEL结构。

表5-25有15个等价变异体，为ASF和ASI两种变异算子，ASF和ASI变异算子所占比例都在11%之上。

表5-24 经验研究的算子分布

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **类型** | **Supply**  **Chain** | **Smart**  **Shelf** | **Supply**  **Customer** | **loan\_**  **approval** | **Car**  **Estimate** | **Travel**  **Agency** | **变异体**  **总计** |
| ISV | 0 | 0 | 0 | 0 | 0 | 6 | 6 |
| EAA | 4 | 0 | 0 | 0 | 0 | 0 | 4 |
| EEU | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| ERR | 20 | 5 | 10 | 10 | 0 | 20 | 65 |
| ELL | 0 | 1 | 1 | 0 | 0 | 0 | 2 |
| ECC | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ECN | 0 | 0 | 0 | 8 | 0 | 0 | 8 |
| EMD | 0 | 0 | 0 | 0 | 2 | 0 | 2 |
| EMF | 0 | 0 | 0 | 0 | 2 | 0 | 2 |
| ACI | 1 | 1 | 1 | 1 | 1 | 1 | 6 |
| AFP | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ASF | 11 | 2 | 7 | 1 | 5 | 6 | 32 |
| AIS | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AEL | 9 | 2 | 2 | 2 | 5 | 7 | 27 |
| AIE | 4 | 1 | 1 | 0 | 0 | 5 | 11 |
| AWR | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| AJC | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| ASI | 10 | 2 | 2 | 1 | 4 | 4 | 23 |
| APM | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| APA | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| XMF | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| XMC | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| XMT | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| XTF | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| XER | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| XEE | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 总计 | 60 | 14 | 25 | 24 | 21 | 50 | 194 |

在精简变异体的时候，需要注意的是等价变异体的存在。分析附录里的26种变异算子，发现AFP、ASF、AIS、ASI可能产生等价变异体。

* AFP算子将forEach活动由顺序改为并行，修改parallel属性由 “no”改为 “yes”，变异体需要更多的时间来完成活动，可能产生等价变异体。
* ASF算子用flow活动替代sequence活动，如果flow活动与sequence活动的结构相同，可能产生等价变异体。
* AIS算子将一个域的isolated属性改为“no”，这意味着这些变量的访问不会被保护，以防止产生没有预想到的结果，可能产生等价变异体。
* ASI算子用一个sequence活动交换两个活动。如果它们之间没有时间依赖关系，它会生成等价变异体。

表5-25 经验研究的算子检测情况

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **类型** | **变异体总计** | **所占百分比（%）** | **等价变异体个数** | **存活变异体个数** |
| ISV | 6 | 3.09 | 0 | 0 |
| EAA | 4 | 2.06 | 0 | 0 |
| EEU | 1 | 0.52 | 0 | 0 |
| ERR | 65 | 33.51 | 0 | 5 |
| ELL | 2 | 1.03 | 0 | 1 |
| ECC | 0 | 0 | 0 | 0 |
| ECN | 8 | 4.12 | 0 | 0 |
| EMD | 2 | 1.03 | 0 | 0 |
| EMF | 2 | 1.03 | 0 | 0 |
| ACI | 6 | 3.09 | 0 | 0 |
| AFP | 0 | 0 | 0 | 0 |
| ASF | 32 | 16.49 | 12 | 0 |
| AIS | 0 | 0 | 0 | 0 |
| AEL | 27 | 13.92 | 0 | 0 |
| AIE | 11 | 5.67 | 0 | 0 |
| AWR | 0 | 0 | 0 | 0 |
| AJC | 1 | 0.52 | 0 | 1 |
| ASI | 23 | 11.86 | 3 | 0 |
| APM | 1 | 0.52 | 0 | 0 |
| APA | 1 | 0.52 | 0 | 0 |
| XMF | 1 | 0.52 | 0 | 0 |
| XMC | 1 | 0.52 | 0 | 1 |
| XMT | 0 | 0 | 0 | 0 |
| XTF | 0 | 0 | 0 | 0 |
| XER | 0 | 0 | 0 | 0 |
| XEE | 0 | 0 | 0 | 0 |
| 总计 | 194 | 100 | 15 | 8 |

在适用的18个变异算子中，ERR、ELL、AJC、XMC生成的变异体没有测试用例能将其杀死，在对BPEL程序进行测试时，需要重点检测这四种错误类型。

ERR算子在6个实例比例高达33.51%。且在各实例里所占的比例都很高，存在5个存活变异体，在BPEL程序测试的时候，对于ERR故障类型（即关系运算符的替换）需要重点检测。

通过前面的6个BPEL程序的变异测试，可以除了ACI算子可以很容易的被检测出来，其他算子的FDR都有一定的浮动。利用5.1提出的算子包含关系，对算子进行的精简总结为以下几点：

1. 对于ACI变异算子，此类型的变异体很容易被杀死，可以精简。
2. 对于AEL和AIE变异算子，在同一测试用例集下，在能杀死MAEL的基础上MAIE都能杀死，即MAEL包含MAIE，相对于AEL算子AIE可以精简。
3. 对于ASI和ASF变异算子，在同一测试用例集下，在能杀死MASI的基础上MASF都能杀死，即MASI包含MASF，相对于ASI算子ASF可以精简。
4. 在带有常量的关系表达式中，ERR相对于ECN是可以精简的。
5. 对于EMD和EMF变异算子，在同一测试用例集下，能杀死MEMD的测试用例都能杀死MEMF，故EMF相对于EMD是可以精简的。
   1. 小结

利用本文开发的MuBPEL系统对SupplyChain、SmartShelf、SupplyCustomer、loan\_approval、CarEstimate、TravelAgency六个BPEL程序进行变异测试，采用经验研究的方式验证了面向BPEL程序的变异测试技术与支持工具的可行性和有效性。通过进一步分析和评估各类型变异体的故障检测率，归纳总结出BPEL程序变异测试时变异测试时变异体精简结论。

6工作总结与展望

BPEL是一种协调多个Web服务完成复杂业务流程的基于XML的语言，由于Web服务松散耦合与开放性，面向BPEL的测试存在一定的难度，变异测试是一种基于故障的测试技术，具有很强的故障检测能力。结合BPEL的特点，本文提出了面向BPEL程序的变异测试技术，开发了面向BPEL程序的变异测试系统MuBPEL，以六个真实的BPEL程序作为实验对象进行经验研究，检验该变异测试技术与支持工具的可行性，同时对变异算子进行分析和评估，寻找不同变异算子模仿的故障之间的检测包含关系，归纳出适合BPEL程序的变异体精简方法。

本文的主要成果如下：

* 提出了一种面向BPEL程序的变异测试框架与过程。
* 开发了面向BPEL程序的变异测试系统MuBPEL，可自动化的生成BPEL程序的变异体，支持BPEL程序的测试执行和测试结果分析，提高了BPEL程序变异测试的自动化。
* 利用六个实例验证了设计的变异测试框架与MuBPEL系统的可行性及有效性。
* 采用经验研究的方式评估变异算子的有效性，实现了变异体的精简。

本文的不足及未来展望：

* 对BPEL程序进行变异测试的实例研究，实现26种变异算子的全覆盖。
* 进一步提高变异体精简效率算子。
* 完善MuBPEL变异测试系统，支持多变异体的批量测试，支持更多类型测试。

参考文献

1. 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(6).
2. [Nikola Milanovic](http://dl.acm.org/author_page.cfm?id=81100389479&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552), [Miroslaw Malek](http://dl.acm.org/author_page.cfm?id=81408602153&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552). Current solutions for web service composition[J]. IEEE Internet Computing, 2004, pp: 51-59.
3. OASIS. 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, 2007.
4. 姚世军, 谢蕾, 吴之铁. 基于WS-BPEL的服务组合流程建模工具的设计与实现[J]. 计算机工程与设计, 2009, (17), pp: 3999-4001.
5. 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(5).
6. 孙喜龙. 基于BPEL的Web服务组合测试研究[D]. 北京工业大学, 2009.
7. Richard A.DeMillo, Richard J. Lipton and Frederick G.Sayward. Hints on test data selection: Help for the practicing programmer[J]. IEEE Computer, 1978(1), pp: 31-41.
8. Antonia Estero-Botaro, Francisco Palomo-Lozano, Inmaculada Medina-Bulo. Mutation Operators for WS-BPEL 2.0[C]. In Proceedings of ICSSEA 2008-2 Estero Palomo Medina, 2008, pp: 1-7.
9. Yu.-Seung Ma, Mary Jean Harrold, Yong-Rae Kwon. Evaluation of Mutation Testing for Object-Oriented Programs[C]. Proceedings of the 28th International Conference on Software Engineering (ICSE’ 06), 2006, pp: 869-872.
10. 刘涛, 高珍, 张志浩. 基于 BPEL4WS 的分布式应用系统的研究与实现[J]. 计算机应用研究, 2004, 21(8), pp: 220-222.
11. 梅彪, 姜新文, 吴恒. WS-BPEL 业务流程与访问控制[J]. 计算机工程, 2008, 34(19), pp: 144-146.
12. 孙昌爱, 王冠. MujavaX:一个支持非均匀分布的变异生成系统[J]. 计算机研究与发展, 2014, 51(4), pp: 874-881.
13. Jia Yue, Harman Mark. An analysis and survey of the development of mutation testing[J]. IEEE Transactions on Software Engineering, 2011, 37(5), pp: 649-678.
14. Chang-ai Sun, Guan Wang, Kai-Yuan Cai, T.Y. Chen. Distribution-aware Mutation Analysis[C]. Proceedings of 9th IEEE International Workshop on Software Cybernetics (IWSC 2012), collocated with COMPSAC 2012, IEEE Computer Society, 2012, pp. 170-175.
15. Boubeta-Puig Juan, Medina-Bulo Inmaculada, García-Domínguez Antonio. Analogies and Differences between Mutation Operators for WS-BPEL 2.0 and Other Languages[C]. Proceedings of 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW 2011), IEEE Computer Society, 2011, pp: 398-407.
16. Estero-Botaro Antonia, Palomo-Lozano Francisco, Medina-Bulo Inmaculada. Quantitative Evaluation of Mutation Operators for WS-BPEL Programs[C]. Proceedings of 3rd International Conference on Software Testing, Verification, and Validation Workshops (ICSTW 2010), IEEE Computer Society, 2010, pp: 142-150.
17. Erich Gamma, Richard Helm, Ralph Johnson, et al. Design Patterns: Elements of Reusable Object-Oriented Software[M]. Addison -Wesley Publising Company, 1995.
18. 杨帆, 王钧玉, 孙更新. 设计模式从入门到精通[M]. 北京: 电子工业出版社, 2010.
19. Sebastian Hinz, Karsten Schmidt, Christian Stahl. Transforming bpel to petri nets[C]. Proceedings of the 3rd International Conference on Business Process Management (BPM 2005), 2005, pp: 220-235.
20. Jesús Arias Fisteus, Luis Sánchez Fernández, Carlos Delgado Kloos. Formal Verification of BPEL4WS Business Collaborations[J]. Lecture Notes in Computer Science, 2004, pp: 76-85.
21. Zhaohong Yang，Yunzhan Gong，Xiao Qing, et al. DTS-A Software Defects Testing System[C]. 2008 Eighth IEEE International Working Conference on Source Code Analysis and Manipulation. IEEE, 2008, pp: 269-270.
22. Bixin Li, Dong Qiu, Shunhui Ji. Automatic test case selection and generation for regression testing of composite service based on extensible BPEL flow graph[C]. Proceedings of IEEE International Conference on Software Maintenance(ICSM 2010), IEEE Computer Society, 2010, pp: 1-10.
23. Dominguez-Jimenez Juan, Estero-Botaro Antonia, Garcia-Dominguez Antonio, et al. GAmera: An Automatic Mutant Generation System for WS-BPEL Programs[C]. 2009 Seventh IEEE European Conference on Web Services(ECOWS '09). IEEE Computer Society, 2009, pp: 97-106.
24. 杨学红, 黄俊飞, 宫云战,等. BPEL 静态缺陷检测方法[J]. 北京邮电大学学报, 2011, 34(2), pp: 108-112.
25. Chang-ai Sun, Yan Shang, Yan Zhao, et al. Scenario-Oriented Testing for BPEL Service Programs[C], Proceedings of 12th International Conference on Quality Software (QSIC 2012), IEEE Computer Society, 2012, pp.171-174.
26. Chang-ai Sun, Yi Meng Zhai, Yan Shang, et al. BPELDebuger: An effective BPEL-specific fault localization framework[J]. Information and Software Technology. Elsevier, 2013, 55(12), pp: 2140-2153.
27. Ma Yu-Seung, Offutt Jeff, Kwon Yong-Rae. MμJava: A Mutation System for Java[C]. Proceedings of the 28th international conference on Software engineering (ICSE’ 06), 2006, pp: 827-830.
28. dom4j 2.0 [EB/OL], http://dom4j.sourceforge.net/, 2010.
29. Wen-Li Dong, Hang Yu, Yu-Bing Zhang. Testing BPEL-based Web Service Composition Using High-level Petri Nets[C]. Proceedings of 8th IEEE International Conference on Enterprise Distributed Object Computing(EDOC’06), IEEE Computer Society, 2006, pp. 441-444.
30. José García-Fanjul, Javier Tuya, Claudio De La Riva, et al. Generating Test Cases Specifications for BPEL Programs of Web Services Using SPIN[C]. Proceedings of WS-MaTe, 2006, pp. 83-94.
31. Jun Yan, Zhong Jie Li, Yuan Yuan Yan, et al. BPEL4WS Unit Testing: Test Case Generation Using a Concurrent Path Analysis Approach[C]. Proceedings of the 17th International Symposium on Software Reliability Engineering(ISSRE’06), IEEE Computer Society, 2006, pp: 75-84.
32. Yuan Yuan，Zhongjie Li，Wei Sun. Graph-Search Based Approach to BPEL4WS Test Generation[C]. Proceedings of IEEE International Conference on Software Engineering Advances(ICSEA’06), IEEE Computer Society, 2006, pp: 14.
33. Yongyan Zheng, Jiong Zhou, Paul Krause Zheng. An Automatic Test Case Generation Framework for Web Services[J]. Journal of Software, 2007, 2(3), pp: 64-77.
34. Sanjai Rayadurgam, Mats Heimdahl. Coverage based testcase generation using model checkers[C]. In Proceedings of IEEE International Conference and Workshops on Engineering of Computer-Based Systems(ECBS), IEEE Computer Society, 2001, pp: 0083.
35. David H. Akehurst. Validating BPEL Specifications using OCL[J]. n nvry, 2004, pp: 2004-2027.
36. [YangChun](http://eprints.qut.edu.au/view/person/Ouyang,_Chun.html) Ou,  [Eric](http://eprints.qut.edu.au/view/person/Verbeek,_Eric.html) Verbeek,  [Stephan](http://eprints.qut.edu.au/view/person/Breutel,_Stephan.html) Breutel, et al. Formal semantics and analysis of control flow in ws-bpel [J]. BPM Center Report. PMcenter.org, 2005.
37. YanPing Yang, QingPing Tan, JinShan Yu, et al. Transformation bpel to cp-nets for verifying web services composition[C]. In Proceedings of the International Conference on Next Generation Web Services Practices, 2005, pp: 137.
38. Michael Butler, Carla Ferreira, Muan Yong Ng. Precise modeling of compensating business transactions and its application to bpel[J]. The Journal of Universal Computer Science, 2005, pp: 712–743.
39. Andreas Wombacher, Peter Fankhauser, Erich Neuhold. Transforming bpel into annotated deterministic finite state automata for service discovery[C]. In Proceedings of ICWS. IEEE Computer Society, 2004, pp: 316.
40. [Hyoung Seok Hong](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Hyoung%20Seok%20Hong.QT.&searchWithin=p_Author_Ids:37347229800&newsearch=true), [Sung Deok Cha](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Sung%20Deok%20Cha.QT.&searchWithin=p_Author_Ids:37289277300&newsearch=true), [Insup Lee](http://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=p_Authors:.QT.Insup%20Lee.QT.&searchWithin=p_Author_Ids:37279665300&newsearch=true). Data flow testing as model checking[C]. In Proceedings of ICSE, IEEE Computer Society, 2003, pp: 232–242.
41. Sandra Rapps, Elaine J. Weyuker. Selecting software test data using data flow information[J]. IEEE Transactions on Software Engineering, 1985, (11), pp: 367–375.
42. [Cesare Bartolini](http://dl.acm.org/author_page.cfm?id=81100312688&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552), [Antonia Bertolino](http://dl.acm.org/author_page.cfm?id=81100192955&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552), [Eda Marchetti](http://dl.acm.org/author_page.cfm?id=81100170657&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552) and [Ioannis Parissis](http://dl.acm.org/author_page.cfm?id=82459139857&coll=DL&dl=ACM&trk=0&cfid=260063615&cftoken=86814552). Data flow-based validation of web services programs: Perspectives and examples [J]. Architecting Dependable Systems V, 2007, pp: 298–325.
43. 孙昌爱. 面向SOA的软件开发中若干开放问题探讨[EB/OL],中国科技论文在线, http://www.paper.edu.cn/index.php/default/releasepaper/content/201107-461, 2011.
44. Chang-ai Sun, Yan Shang, Yan Zhao, et al. Scenario-Oriented Testing of Service Programs using BPEL[C]. Proceedings of 12th International Conference on Quality Software (QSIC 2012), 2012, pp: 171-174.
45. Hazlifah Mohd Rusli, Mazidah Puteh, Suhaimi Ibrahim, et al. A comparative evaluation of state-of-the-art web service composition testing approaches[C]. Proceedings of 6th International Workshop on Automation of Software Test (AST 2011), 2011, pp: 29-35.

附录 面向BPEL2.0的变异算子

|  |  |
| --- | --- |
| **标识符替换变异算子（Identifier replacement operators）** | |
| ISV | 用一个相同类型的变量标识符替换另一个 |
| **表达式变异算子（Expression operators）** | |
| EAA | 用一个相同类型的运算符对算术运算符(+，-，\*，/)进行替换 |
| EEU | 移除所有表达式中的一元减法运算符 |
| ERR | 将关系运算符(<， >，<=， >=， =， !=)用另一个相同类型的运算符进行替换 |
| ELL | 将逻辑运算符(and，or)用另一个相同类型的运算符进行替换 |
| ECC | 将路径运算符(/，//)用另一个相同类型的运算符进行替换 |
| ECN | 增加或减小一个单元中数字常量的值，添加或移除一个数字 |
| EMD | 将持续时间用0或者是其一半代替 |
| EMF | 将截止时间用0或者是其一半代替 |
| **活动变异算子（Activity operators）** | |
|  | **与并发相关** |
| ACI | 将createInstance属性从“yes”变为“no” |
| AFP | 将顺序的forEach活动改为并行的 |
| ASF | 用flow活动替换sequence活动 |
| AIS | 将一个域的isolated属性改为“no” |
|  | **与并发无关** |
| AEL | 删除一个活动 |
| AIE | 移除if活动中的elseif或else元素 |
| AWR | 用一个repeatUntil活动替换while活动 |
| AJC | 移除joinCondition元素 |
| ASI | 交换两个sequence孩子活动的序列 |
| APM | 移除pick活动的onMessage元素 |
| APA | 从pick活动或者事件处理程序中删除onAlarm元素 |
| **异常与事件变异算子（Exception and event operators）** | |
| XMF | 移除错误处理程序中的catch或catchall元素 |
| XMC | 移除补偿处理程序的定义 |
| XMT | 移除终端处理程序的定义 |
| XTF | 用throw活动替代抛出故障 |
| XER | 移除rethrow活动 |
| XEE | 从事件处理程序中移除onEvent元素 |

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 |
| 2008.9-2012.6 | 在北京交通大学算机科学与技术攻读 |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. 在学期间从事的科研工作

* 中央高校基本科研业务费资助项目“面向SOA的新型软件测试技术与工具”（FRF-SD-12-015A），主要参与人。

1. 在学期间所获的科研奖励
2. 在学期间发表的论文
3. 胡荣,**王巧玲**,孙昌爱,等. MuBPEL:一个面向BPEL的变异体自动生成系统 [EB/OL],中国科技论文在线, http://www.paper.edu.cn/releasepaper/content/201411-445, 2014.

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** |
| BPEL，变异测试，变异算子，MuBPEL，精简技术 | 公开 | TP311 | 004.41 | 中央高校基本科研业务费资助项目“面向SOA的新型软件测试技术与工具”(FRF-SD-12-015A) |
| **学位授予单位名称\*** | | **学位授予单位代码\*** | **学位类别\***  **工学** | **学位级别\***  **硕士** |
| 北京科技大学 | | 10008 | 工学 | 硕士 |
| **论文题名\*** | | **并列题名** | | **论文语种\*** |
| 面向BPEL程序的变异测试技术与支持工具研究 | |  | | 中文 |
| **作者姓名\*** | 王巧玲 | | **学号\*** | S20121223 |
| **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\*** |
| 计算机科学与技术 | | 软件测试 | 2.5年 | 2015 |
| **论文提交日期\*** | 2012.12.22 | | | |
| **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** | |
| 王成耀（校内）  赵方（校外） | 胡长军 | | 王成耀  张桃红 | |
| **电子版论文提交格式** 文本（pdf） 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\*** | 61 | | | |
| 共33项，其中带\*为必填数据，为22项。 | | | | |