**北京科技大学**

**硕士学位研究生**

**选题报告及文献总结**



**基于切片的BPEL程序故障定位技术与工具研究**

指导教师： 孙昌爱

单 位： 计算机与通信工程学院

学 号： S20140932

姓 名： 冉玉凤

专业名称： 软件工程

入学时间： 2014年9月1日

2015年09月09日

目录

1选题的背景和意义 3

1.1选题的背景 3

1.2 BPEL程序的特点 3

1.3 选题的意义 4

2国内外研究现状 5

2.1国外研究现状 5

2.2国内研究现状 7

2.3 课题组研究现状 7

3本文研究内容 8

3.1基于切片的BPEL程序故障定位技术 8

3.2实例验证 9

3.3基于切片的BPEL故障定位技术开发工具 10

3.4技术难点 11

3.4.1测试用例执行路径和变量的获取 11

4研究工作进度安排 12

4.1工作计划 12

4.2最终成果形式 12

5参考文献 13

# 1选题的背景和意义

## 1.1选题的背景

程序调试就是开发人员找到程序中的错误并修正错误[19]，当程序运行的输出结果和预期的输出结果不一样时，说明程序出现了错误。这时就需要程序调试人员进行调试程序。通常，程序员或者是手工的进行程序错误的定位，或者是利用调试工具，一般过程是设置断点，运行程序查看结果是否与预期的相符，若一样则说明该断点部分没有错误，否则，表示该断点处有错误，最后是定位错误。这一过程不断地进行，直至程序运行的结果和预期结果一样，调试过程结束。由此可以看出软件调试是软件开发和软件的后期维护中比较繁琐和耗费时间及精力的工作。因而越来越多的人希望能够找到一种技术来自动化定位程序中存在的错误。

在传统的软件调试技术中，针对不同的语言也有不同的调试方式。针对新出现的编程语言 BPEL(Business Process Execution Language)，是一种基于XML的自动化业务流程语言，广泛运用于Web服务相关的项目开发中[12]。但是到目前为止，对于BPEL程序的故障定位技术依然很少。因此拟提出了一种基于切片的BPEL程序故障定位技术。

程序切片是一种程序分析和理解技术[25]。它通过把程序减少到只包含某个特定计算相关的那些语句来分析程序，程序切片的基本思想是：程序的某一个输出只与源程序中部分语句和控制谓词有关，删除其它的语句和谓词并不影响该输出的结果。这也表明，对于该输出来说，源程序与删除不相关语句和谓词以后所得的可执行程序在语义上是一致的。

基本的程序切片可分为静态切片和动态切片[33]，静态切片不需要执行程序，即可找到影响变量值的语句；而动态切片依赖程序的特定输入，保留对最终结果有影响的语句。通过运行动态切片，有助于找出错误，缩小错误的范围，广泛应用于程序的调试。

## 1.2 BPEL程序的特点

BPEL是一种基于XML的构建在Web服务上的编程语言[26]，BPEL程序本身是一个服务，只不过他的功能是将已经定义过的具体的服务组装起来，因此在BPEL的定义中使用了WSDL的描述。

BPEL程序模块被划分为原子语句块和非原子语句块，原子语句块是指原子执行步骤，包括assign、invoke、receive、reply等操作，非原子语句块是指组合语句块，包括sequence、switch、while、flow等。在非原子语句块中其实是有多个原子语句块组成。由于BPEL程序有flow（并发）模块，所以BPEL程序区别于传统的程序，具有多分支多并发的特点。因而当一个测试用例输入时，有多条执行路径。

在整个BPEL程序过程中，首先通过接受(Receive)操作来接收传入的参数。在接收到参数以后，BPEL自身会通过各种赋值 (Assign) 操作来决定数据如何流动，并且通过唤醒(Invoke)操作来调用相应的Web服务。整个BPEL就是定义如何通过它的接口输入参数，调用其他外部服务的接口，得到其返回值的过程。因而有着大量的变量的定义和变量之间数据的传递。其终点就是回应(Reply)操作，也就是将程序的结果进行返回。

## 1.3 选题的意义

常用的程序切片技术，是对整个程序进行程序切片，造成程序切片过大，不能准确的定位故障。本文提出的基于切片的BPEL故障定位技术，它通过依赖关系将程序中的切片分为数据切片和谓词，首先通过对谓词的切换找到关键谓词。然后，寻找关键谓词的后向切片，根据后向切片与谓词的关系来进行错误的定位，实现了减小切片大小的效果。而且所描述的这种方法在C语言调试上取得了很好的效果，这种调试方法的优点在于：

（1）基于谓词的调试方法将错误输出语句的测试用例执行路径与预期路径进行对比，找到关键谓词，它减少了程序中语句的分析数量，并且得到的语句集合包含错误的可能性极大的增加，降低了故障定位的难度。

（2）基于切片的调试方法应用于BPEL程序，鉴于BPEL程序的树形结构，任意一个测试用例有且仅有一条执行的路径，如果产生错误，只用在一条路径中搜索，通过寻找该条路径中的关键谓词与数据切片的关系，大大的提高了故障定位的效率。

BPEL程序作为一种后起的编程语言，在Web服务开发过程中发挥着日益重要的作用，软件的规模和层次都不断的在增加调试的难度，程序的自动化调试也越来越受到研究者们的重视。如果能够将基于切片的精确故障定位调试方法成功的应用于BPEL程序的调试过程中，对整个Web服务开发的巨大贡献。

# 2国内外研究现状

## 2.1国外研究现状

程序切片该概念由Weiser于1979年在其博士论文中首次提出[5]，程序切片是一个可执行程序，切片与源程序对某个特定变量来说，在程序特定位置语义上应是一致。一般来说，一个程序切片对应一个特定的切片准则(slicing criterion)，记为<s，v>，其中s为源程序中的某一条语句，v为语句s处定义或引用的变量，程序切片可以通过从源程序中删除零条或多条语句得到，但要保证源程序和切片关于切片准则<s，v>的行为相同。Weiser提出的程序切片方法属于静态程序切片技术。

Horwitz通过将程序依赖图扩展为系统依赖图[22]，从而得到过程间切片；Korel和Laski引入动态切片[7,8]，主要寻找程序某次执行中影响变量在程序中某点状态的语句；Kamkar、Shahmehri和Fritzson提出在执行过程中构造动态依赖图生成动态切片；Agrawal、Demillo和Spaffford提出依赖图算法[25]将定义基于存储单元的定义和引用上，而Duesterwald、Gupta和Soffa提出依赖图算法用切片中决定信息陈述代替程序中非决定信息陈述。

Korel和Laski提出了动态切片[7,8]的概念，并将动态切片分为了两大类：后向动态切片和前向动态切片，后向动态切片包括了在已经执行过的路径中与受影响的变量相关的所有程序切片。而前向动态切片和后向动态切片刚好相反，前向动态切片是指出现在影响点之前的所有和影响点中的变量相关的程序切片，在程序的执行过程中是从后向切片向前向切片逐渐推进的，研究后向切片有助于帮助程序员来调试程序。

在对后向切片的研究中，Agrawal和DeMillo[23]提出了一种调试的模型，主要思想就是：如果发现程序E{e1,e2,e3,ek,……en}在某个点ek处发生了错误的输出或者出现了错误导致了程序的崩溃，这里的ek通常具有不确定性，这个主要根据文中定义的一套规则来选择，一般来讲涉及到的变量的数量越多越容易发生错误，则该点的切片则作为优先考虑的对象。首先，在模型中需要找到ek的后向切片，如果与ek点相关的变量在后向切片中只出现一个，则只需要找到和ek相关的变量即可，如果与ek相关的变量在后向切片中有多个变量{v1,v2,vk……vn}，则需要分别去找到这些变量的切片集合P{{ (e1,v1)}, ,{ (ek,vk)}……{ (en,vn )}}。然后每次针对一组切片进行处理，将重新获取的结果和预期结果比对看是否相等来定位错误的位置。

后向切片的研究对于程序错误的定位固然重要，但是对于一个规模较大的程序，如果错误语句出现的位置比较靠近程序的结尾，那么其后向切片的规模也是相当庞大的，要在这庞大数量的切片中去找到错误的根源实属不易，况且在庞大的后向切片中存在大量的与错误切片毫无关系的切片，不仅干扰了错误的查找，而且增加了定位的难度。针对此问题文章[24]提出了错误诱导的排查方法，旨在减小与错误代码相关的切片的范围。文章首先利用Zeller提出的Delta Debugging[12]方法，Delta Debugging是一项自动为计算机程序分析出失败原因的技术，同时也将他和Hildebrandt共同提出的delta方法[4]用于错误诱导切片的决策中，这种方法其实是Delta Debugging和Dynamic slicing的整合。首先，通过Delta Debugging算法来得到最小错误诱导输入，然后在这个输入的基础上计算出前向动态切片并与以错误诱导准则产生的动态后向切片的交集，得到引发故障的片段(chop)，实验证明，和庞大的动态切片相比，这种片段(chop)的数量更小，包含错误的能力也很强。但如果修改集中没有产生错误的语句，而错误是由其后向切片所产生，并不能精确地找到产生错误的位置。

## 2.2国内研究现状

文献[31]提出了基于程序依赖图的静态BPEL程序切片技术，传统的程序切片技术在计算BPEL程序切片时会产生切片不完备问题，提出一种基于程序依赖图的BPEL静态切片技术，通过建立BPEL程序依赖图，计算BPEL的程序切片。但是由于静态切片和程序的不受程序的输入影响，产生的切片过大。

文献[29]提出了业务流程切片的概念和对应的静态切片算法。流程切片是根据给定的活动和活动引用的变量(称为切片准则),分析出在整个流程中能够影响切片准则的流程片段。介绍了静态切片算法在给定切片准则的前提下,首先构造BPEL控制流图,在此基础上生成活动依赖图,并对活动依赖图进行分析,从而确定流程中影响切片准则的活动序列(流程片段)。

## 2.3 课题组研究现状

课题组在前期的研究工作中已经取得了一些成果[21][33]，主要是将基于频谱的故障定位技术用于BPEL程序的故障定位中，这种方法的主要思想[33]是在把程序划分为多个模块(Block)的基础上，通过大量的测试用例来计算每一个测试点的怀疑度，在计算怀疑度的过程中首先得将所有的测试用例的测试结果成功(success)或者失败(failed)记录下来，同时获取到每一个测试用例的测试路径，即该测试用例经过了哪些模块。怀疑度的计算公式在多种故障定位方法中都有提到，如Tarantula[9]方法、Set-Union[9]方法、Code Coverage[19][20]方法等,然后将怀疑度从高到低进行排序，一般来讲，怀疑度越高的语句所包含的错误的能力就越强，在前期的研究中主要是通过多种不同的方法计算在BPEL程序中多个被测点可疑度来得到可疑度最大的程序片段，这种故障定位方法的有效性可以达到50%左右。

实验室已经提出一种基于谓词切换的BPEL调试技术，首先要将BPEL程序划分成多个模块(Block)，然后找到程序中所有的谓词所在的模块，通过对谓词不断的切换并对比实际输出与预期输出的结果以确定关键谓词，找到关键谓词以后顺着程序执行的路径进行后向分析。通过遍历查询每个模块的变量信息来获取相关联的模块。

# 3本文研究内容

## 3.1基于切片的BPEL程序故障定位技术

根据BPEL语言的特点，提出一种基于切片的BPEL精确故障定位技术，从而可以帮助软件工程人员更好地调试BPEL程序。本文提出的切片技术与实验室前期的基于谓词切换的故障定位方法相结合，能够更精确的定位故障位置

基于切片调试方法应用于BPEL程序的调试中区别于传统程序的调试，首先要将BPEL程序划分成多个模块(Block)，然后找到程序中所有的谓词所在的模块，通过对谓词不断的切换并对比实际输出与预期输出的结果以确定关键谓词，找到关键谓词以后顺着程序执行的路径获取谓词的后向数据切片，分析谓词与数据切片之间的关系，从而定位错误的根源。在BPEL程序中摒弃了传统程序中有关函数、断点的考虑，仅需考虑整个程序的逻辑流程，探索到关键谓词，然后在堆栈的基础上探求每一个数据切片在该执行路径上的动态变化，进行综合的分析即可得知错误的程序模块。

本文的主要研究内容就是将基于切片的调试方法应用于BPEL程序的调试过程，调试的前提是需要通过大量的测试用例来发现程序中的错误。

　　根据BPEL程序的特点，本文所探索的定位方法大致可以分为以下步骤：

步骤1：对事先拟好的测试用例进行输入，并记录每一个测试用例ti的实际测试结果及通过的测试路径所对应的Block块bps={ sk,……sj,sn},其中bps∈bp。

步骤2：针对于每一个测试用例ti的实际输出结果与预期输出进行比较，即可得出每一个测试用例的pass或者failed状态。

步骤3：对于所有的failed状态的测试用例的集合cs=｛c1,c2,……,cn｝中的每一个测试用例ci，获取执行路径中的谓词集合pres={ p1,p2,……,pn }∈bp,分别更改每一个谓词pi的逻辑值，强制改变判断分支，并记录下该结果的pass与failed状态,在执行完ci后会出现两种情况：

（1）执行完ci后，结果的状态为pass，就找到了关键谓词。

（2）执行完ci后，结果的状态为failed，则继续寻找下一个谓词。

步骤4：根据关键谓词，来寻找两个谓词之间的后向数据切片，首先找出谓词中的变量，然后找出变量的数据依赖。根据后向数据切片S1中的值，来确定错误语句的模块。而对于错误输出的语句，进行后向数据切片S2。S1和S2的并集作为切片集合St，即同时考虑了错误的输出语句的控制依赖和数据依赖。

采用的是一种基于动态流的程序切片方法，首先提取出对应给定历史节点的数据依赖点，定义出历史记录的变量的定义-引用依赖关系，然后通过递归算法，获得关键谓词的动态切片，从而达到故障定位的效果。

## 3.2实例验证

在前期的工作中，我们使用基于频谱的故障定位方法来对BPEL程序进行故障定位，该定位方法的定位正确率可以达到约50%。通过基于谓词切换的故障定位方法能够有新的突破。这里通过smartshelf例子介绍基于切片的BPEL程序的故障定位方法的应用。

首先是错误的植入，将事先设计的各种错误版本按顺序植入到smartshelf的BPEL程序中，根据方法的要求每次只能植入一个错误，否则该方法无效。

其次，根据程序中的谓词，选择测试用例，测试用例应该尽可能的去覆盖程序中的谓词所在的分支，所选择的测试用例样式如下：candy&&100，cookie&&200，milk&&100等。根据smartshelf例子结构特点，每一个测试用例都会至少经过一个谓词。

再次，针对每一个测试用例，首先要记录在错误版本中的输出结果，而且还要记录在错误版本中的分支，分支用模块(Block)序号表示。例如，candy&&100的输出结果是

Quantity is sufficient!

Location is right!

[SendStatusToWarehouseWebService]: Expired products has been replaced!

而预期的输入是：

Quantity is sufficient!

Location is right!

Goods are in good status!

因为实际结果和预期结果存在误差，说明程序中出现了错误，candy&&100所经历的测试路径是：p{1-15,18-21,27-32,50-53}，在该条路径中的谓词有（$\_status != 0）、（$\_location = 0）、（$\_amount < $init\_amount）对该谓词进行排序后对每一个谓词进行更改，修改第一个谓词之后得出的结果和预期结果相等，那么该谓词为关键谓词。

然后对变量$\_status进行后向的数据切片，找出与谓词中变量相关的程序块，得到该变量与数据的输入有关。

## 3.3基于切片的BPEL故障定位技术开发工具

该工具的设计主要来源于基于切片的调试的思想，因此在工具设计的过程中得包含以下功能：BPEL程序的解析功能，发送消息功能，执行层,图形界面层及辅助工具层。

第一，在工具中首先需要对BPEL程序进行解析，以此来划分出不同的模块，因此在解析功能中主要提供两种服务，第一种是BPEL程序的解析功能，另外一种是根据节点的name属性获取任意节点并修改其条件值的功能。

第二，针对一套完整的服务流程，我们需要对其发送消息，并通过结果检测来分析程序的错误，我们需要一个客户端对BPEL的流程服务发送消息，该消息的发送是基于SOAP协议的消息。发送消息层提供的主要服务就是对整个BPEL流程服务发送SOAP消息，在发送消息的过程中需要等上一个消息结果返回以后才开始发送下一个消息，中间需要提供一段睡眠时间。同时，BPEL流程服务将最终的结果以SOAP格式进行返回。

第三，发送消息之后，需要对结果进行解析，满足后续工作的需要，将每一个测试用例对应的实际输出结果和预期的结果进行比对，记录测试用例的成功(passed)或者失败(failed)状态。另外,一个很重要的功能也需要在测试用例的过程中被同步记录，那就是用例执行的路径。因此在执行层主要提供这三种服务，即：测试结果解析、用例的比较和路径的记录。

第四，根据路径的记录，找到关键谓词，关键谓词的查找过程实验室已经实现，现在只需针对关键谓词进行后向数据切片，找到关键谓词中的变量，对此变量查找其定义使用关系，最终找到关键谓词的后向数据切片。

第五：在使用工具的过程中用户不可能永远在控制台进行输入输出，而且在控制台输入输出没法体现整个工具的完整性，因此加入了相关的图形界面使整个调试的流程通过界面的效果得以体现。另外，程序中涉及的一些其他的小功能主要放在util包中，辅助其它功能的完成。

## 3.4技术难点

### 3.4.1测试用例执行路径和变量的获取

在本次试验中主要有三个大的模块， BPEL程序解析、发送消息、程序执行过程中的信息获取模块，目前，已经实现了前两个模块，在第三个模块中最棘手的问题是：

1. 对单个的测试用例的执行路径的获取。
2. 确定变量的数据依赖关系。

# 4研究工作进度安排

## 4.1工作计划

|  |  |
| --- | --- |
| 时间段 | 任务 |
| 2015.6－2015.8 | 收集各种资料并阅读文献。 |
| 2015.9－2015.12 | 研究测试用例路径获取的方法，如何在用例执行的过程中解析出其经过的路径和执行过程中的变量信息。 |
| 2016.1－2016.4 | 研究ODE的引擎，对其数据依赖，进行分析。 |
| 2016.5－2016.6 | 根据动态切片的算法对BPEL进行动态切片，生成工具。 |
| 2016.7－2016.8 | 调试工具，进行实验验证。 |
| 2016.9－2016.10 | 撰写论文，准备答辩。 |

## 4.2最终成果形式

（1）完成BPEL的后向程序切片算法；

（2）完成基于切片的BPEL程序故障定位技术的支持工具；

（3）力争在较有影响力的期刊或者会议上发表一至两篇论文。

# 5参考文献

1. Yuan Y, Li Z, Sun W. A graph-search based approach to BPEL4WS test generation[C]//Software Engineering Advances, International Conference on. IEEE, 2006: 14-14.
2. Barreto C, Bullard V, Erl T, et al. Web services business process execution language version 2.0[J]. Specification, OASIS, 2007.
3. Li L, Chou W, Guo W. Control flow analysis and coverage driven testing for web services[C]//Web Services, 2008. ICWS'08. IEEE International Conference on. IEEE, 2008: 473-480.
4. Zeller and R.Hildebrandt. Simplifying and isolating failure-inducing input, IEEE Transactions on Software Engineering(TSE), Feb,2002,28(2).
5. Ohata F, Hirose K, Fujii M, et al. A slicing method for object-oriented programs using lightweight dynamic information[C]//Software Engineering Conference, 2001. APSEC 2001. Eighth Asia-Pacific. IEEE, 2001: 273-280.
6. Weiser M.Program slicing [J].IEEE Trans Software Eng,1994.10(4):352-357
7. Korel B,Laski J.Dynamic slicing of computer programs[J].J Systems and Software,1990,13(3):187-195.
8. Korel B, Rilling J.Dynamic program slicing methods [J]. Information and Software Technology,1998,40(11):647-659.
9. J.A.Jones and M.J.Harrold. Empirical Evaluation of the Tarantula Automatic Fault-Localization Technique[C],Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005),2005:273-282.
10. Moser S, Haebich M, Martens A. Generating compatible partner processes in BPEL: U.S. Patent 8,914,770[P]. 2014-12-16.
11. Palomo-Duarte M, García-Domínguez A, Medina-Bulo I. Automatic dynamic generation of likely invariants for WS-BPEL compositions[J]. Expert Systems with Applications, 2014, 41(11): 5041-5055.
12. Zeller.Yesterday, my program worked. Today, it does not. Why?,Seventh European Software Engineering Conference/Seventh ACM SIGSOFT Symposium on Foundations of Software Engineering(ESEC/FSE),Sept 1999: 253-267.
13. Elser J J, Bracken M E S, Cleland E E, et al. Global analysis of nitrogen and phosphorus limitation of primary producers in freshwater, marine and terrestrial ecosystems[J]. Ecology letters, 2007, 10(12): 1135-1142.
14. Palomo-Duarte M, García-Domínguez A, Medina-Bulo I. Improving Takuan to analyze a meta-search engineWS-BPEL composition[C]//Service-Oriented System Engineering, 2008. SOSE'08. IEEE International Symposium on. IEEE, 2008: 109-114.
15. Wang H D, Xing J C, Song W, et al. Static BPEL program slicing technique based on BPEL program dependence graphs [J][J]. Journal of Computer Applications, 2012, 8: 066.
16. Bao L, Song S, Chen S, et al. Research on static process slicing in BPEL[J]. Systems Engineering and Electronics, 2009, 31(1).
17. BAO L, CHEN S, SONG S, et al. Research on the static optimization algorithm for the BPEL process[J]. Journal of Xidian University, 2010, 1: 008.
18. OASIS Standard.Web Service Business Process Execution Language Version 2.0.http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html,2012
19. W. E. Wong, Y. Qi, L. Zhao and K. Y. Cai. Effective Fault Localization using Code Coverage[C],Proceedings of 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), 2007:449-456.
20. L. Zhao,Z. Zhang, L.Wang,and X. Yin.PAFL: Fault localization via noise reduction on coverage vector[C], Proceedings of the 23th International Conference on Software Engineering and Knowledge Engineering(SEKE 2011), 2011:203-206.
21. Chang-aiSun,YiMengZhai,YanShang.BPELDebugger:An effective BPEL-specific fault localization framework.Information and Technology, 2013,55(12):2140-2153.
22. S.Horwitz,T.Reps,and D.Binkley. Interprocedural slicing using dependence graphs,ACM Transaction on Programming Languages and System, Jan1990,12(1):26-60.
23. H.Agrawal,R.DeMillo,and E.Spafford. Debugging with dynamic slicing and backtracking, Software Practice and Experienc 1993,23(6) :589-616.
24. Neelam Gupta and Haifeng He, Locating Faulty Code Using Failure- Inducing Chops,ACM International Conference on Automated Software Engineering .Long Beach,CA·USA·2005:263—272.
25. Tip F. A survey of program slicing techniques[J]. Journal of programming languages, 1995, 3(3): 121-189.
26. MAO CHENGYING． Slicing Web service- based software ［C］/ / IEEE

International Conference on Service- Oriented Computing and Applica-

tions． Piscataway: IEEE， 2009: 1 －8．

1. 宋巍,唐金辉,张功萱,等.WS-BPEL服务可替换性分析［J］中国科学: 信息科学,2012,42(3): 264－279.
2. 李必信． 程序切片技术及其应用［M］． 北京: 科学出版社，2006．
3. 鲍亮,宋胜利.BPEL静态流程切片技术研究.系统工程与电子技术.2009,31(1):241-249.
4. 黄亮,姚放吾.Apache ODE环境下Web服务组合技术的研究.计算机技术与发展,2011,21(7):98-104.
5. 王洪达,邢建春,宋巍,杨启亮.基于程序依赖图的静态BPEL程序切片技术.计算机应用.2012,32(8):2338-2341.
6. 黄俊飞,杨红学,宫云战.BPEL流程间死锁检测研究.计算机学报. 10.3724/SP.J.1016.2011.02427
7. 翟忆蒙,面向BPEL程序的故障定位方法与工具研究[D].北京:北京科技大学计算机与通信工程学院,2014.
8. 张东源.软件测试中的程序切片技术,现代导航,2012,03(6):231-238.
9. 曹鹤玲,姜淑娟,鞠小林,基于动态切片和关联分析的故障定位方法[J].计算机学报,2013,37(11):104-109.