计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.200900051

一种基于变异分析的 BPEL 程序故障定位技术

孙昌爱　张守峰　朱维忠
北京科技大学计算机与通信工程学院　北京 100083

摘　要　不同于传统 C,C++或 Java 程序,BPEL (Business Process Execution Language)程序由一组活动及其之间的交互组成,同时引入了并发、序列化、XML 表示等新特征,这些新特点使得定位 BPEL 程序的故障具有一定的挑战性。针对现有故障定位技术在有效性方面的不足,提出一种基于变异分析的 BPEL 程序故障定位技术,依据 BPEL 程序的特点及其变异算子的特点设计了一组优化策略,开发了相应的支持工具。通过一组 BPEL 程序实例来评估所提方法的有效性,比较了所提方法与现有 BPEL 程序故障定位技术的定位效果。实验结果表明,与现有方法相比,所提方法具有较高的召回率,故障定位代价基本相当,提出的优化策略进一步降低了所提方法的变异执行开销。

关键词:故障定位;程序调试;BPEL;变异分析;软件测试
中图法分类号　TP311

Mutation Based Fault Localization Technique for BPEL Programs

SUN Chang-ai,ZHANG Shou-feng and ZHU Wei-zhong
School of Computer and Communication Engineering,University of Science and Technology Beijing,Beijing 100083,China

Abstract　Unlike traditional C,C++,or Java programs,BPEL (Business Process Execution Language) programs are composed of a set of activities and their interactions,which have the new features such as concurrency,synchronization,and XML-based representation. These new features pose difficulties for effectively locating faults in BPEL programs. To address the limited effectiveness of existing fault localization techniques,we propose a mutation-based BPEL program fault localization technique,design a set of optimization strategies based on characteristics of BPEL programs and their mutation operators,and develop a supporting tool. 6 real-life BPEL programs are conducted to evaluate the feasibility and fault localization effectiveness of the proposed technique and its effectiveness is also compared with that of a set of benchmark techniques. Experimental results show that the proposed technique has a higher recall rate while a comparable cost is compared with benchmark techniques,demonstrating that the proposed optimization strategies reduce the mutation cost of the proposed technique.

Keywords　Fault localization,Program debugging,BPEL,Mutation analysis,Software testing

1　引言

计算机软件已经渗透到现代社会的方方面面,与此同时,航空、核能等安全攸关领域频繁出现软件故障,不仅导致了严重的经济损失,甚至可能造成生命威胁。软件测试是常见的质量保证手段,其目的是尽可能多地检测出程序中潜藏的故障。故障定位是程序调试的关键,其目的是确定故障的位置。当程序规模较大时,如何定位故障成为一项繁琐、耗时且昂贵的活动[1]。

近年来,面向服务的架构(Service-Oriented Architecture,SOA)正成为互联网环境下的软件开发新范型[2],被广泛用于开发各种分布式程序。Web 服务是一种遵循 SOA 的应用单元,可以将多个 Web 服务以服务组合的方式协调起来执行复杂的业务流程。BPEL 是一种被广泛认可的 Web 服务组装语言。与传统 C,C++,Java 程序不同的是,BPEL 程序通常表示为 XML 文件,且引入了并发、序列化等新特点。针对新型 BPEL 程序故障定位问题,我们提出了基于语句块的故障定位技术[3]、基于谓词切换的故障定位技术[4]、基于谓词切换与程序切片相结合的故障定位技术[5]等。尽管这些工作较为系统地探究了 BPEL 故障定位问题,但是故障定位召回率和定位精度仍有待进一步提高。

基于变异分析的故障定位方法[6]通过程序变异引入各种可能的故障类型,计算并排序每个语句的怀疑度,按照怀疑度高低推荐故障位置,实验结果表明该方法有良好的故障定位效果。本文研究基于变异分析的 BPEL 程序故障定位技术,并依据 BPEL 程序及其变异算子的特点对该技术进行优化。在此基础上开发相应的支持工具,采用一组 BPEL 程序实例评估所提技术的有效性以及优化策略的效果。

到稿日期:2020-09-04　返修日期:2020-10-13　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61872039);中央高校基本科研业务费专项资金资助项目(FRF-GF-19-019B)
This work was supported by the National Natural Science Foundation of China(61872039) and Fundamental Research Funds for the Central Universities(FRF-GF-19-019B).
通信作者:孙昌爱(casun@ustb.edu.cn)

2 背景知识

与传统的C,C++,Java等程序不同的是,BPEL程序是一种工作流程序,由一组活动及其活动间的不同关系组成。BPEL程序有如下特点[3]:1)表示为XML文件,其语法格式与传统程序差别大;2)参与组合的Web服务可以由不同的编程语言实现;3)支持并发与序列化等机制;4)引入了伙伴链接、流程和补偿处理等新元素。

变异测试(也称变异分析)是一种基于故障的软件测试技术[6],被广泛用于评估测试用例集的充分性与测试技术的有效性。变异分析通过构造人工故障来模拟程序员编程时可能出现的错误,采用给定的测试用例集测试植入故障的程序,通过变异得分来衡量测试用例集的故障检测能力。具体来说,即:将程序中典型错误的模拟称作变异算子;将利用变异算子得到植入故障程序的过程称为变异;将存在故障的程序称为变异体。如果某个测试用例的输出结果与预期结果不同,则该变异体被杀死;一个测试用例集的变异得分定义为能够杀死的变异体与全部变异体的比例。

近年来,人们将变异测试应用于BPEL程序中,评估面向BPEL程序测试技术的故障检测能力[7],提出了面向BPEL的变异测试框架[8-9]。针对BPEL程序的新特点,Estero-Botaro等归纳了34种BPEL变异算子[10],这些变异算子可以模拟常见的BPEL程序故障类型,包括标识符替换、表达式替换、活动替换、异常与事件替换。图1给出了一个BPEL程序的ASF类型故障。在该故障实例中,flow活动被sequence活动替换。本文将基于上述面向BPEL程序的变异测试框架与变异算子,探索BPEL程序的故障定位问题。

```
〈bpel:sequence name="main"〉
  〈bpel:receive name="receiveInput" partnerLink="client"〉
  〈bpel:assign name="a"〉…〈/bpel:assign〉
  …
  〈bpel: sequence name="b"〉 /* 正确的应为 flow */
    〈bpel:invoke name="c"〉…〈/bpel:invoke〉
    〈bpel:invoke name="d"〉…〈/bpel:invoke〉
  〈/bpel: sequence 〉 /* 正确的应为 flow */
  …
  〈bpel:reply name="replyOutput",…/〉
```

图1 BPEL程序ASF类型故障示例

Fig. 1 Example of ASF type fault in a BPEL program

3 基于变异分析的BPEL程序故障定位方法

本节首先描述基于变异分析的BPEL程序故障定位方法的框架及其关键问题,然后描述示例方法的应用,最后讨论优化策略。

3.1 方法框架

基于变异分析的故障定位技术(简称MBFL[6])利用变异算子对给定程序植入故障生成变异体,使用测试用例集分别执行给定程序与变异体,通过变异体被杀死与否的信息来计算程序各语句的怀疑度。当采用MBFL进行BPEL程序故障定位时,首先生成满足一定覆盖准则的测试用例集,其次对给定程序的各语句进行变异操作,得到相应的变异体集,最后采用某个统计公式计算各语句的怀疑度。

但是,BPEL程序的一些新特点使得上述MBFL无法直接实施。因为BPEL程序由原子活动和结构化活动构成,每个活动由多条语句组成,所以BPEL变异算子相应地是针对BPEL程序中的各活动进行的变异操作,无法针对单条语句进行BPEL程序的故障定位。鉴于我们在前期研究工作中提出的基于块的BPEL程序故障定位框架[3],本文提出一种基于变异分析的BPEL程序故障定位框架,如图2所示。该框架将BPEL程序分为原子语句块和非原子语句块,首先对待测BPEL程序的各语句块进行变异操作,得到变异体集合,然后执行待测程序与变异体集合,最后利用执行结果信息计算各语句块的怀疑度。该框架包括调试执行、变异体生成与执行、怀疑度计算等3个主要步骤,具体介绍分别如下:

(1)调试执行:采用测试用例集运行待测程序,记录各个测试用例的执行结果,获取各个测试用例的待测程序语句块的覆盖信息。

(2)变异体生成与执行:使用变异算子在语句块中植入故障得到变异体集合,采用测试用例集运行变异体集合中的所有变异体,获取变异体执行结果及变异体是否被杀死的信息。

(3)怀疑度计算:针对每个变异体的执行信息,结合测试执行阶段的执行结果(即测试用例通过或不通过的信息),采用某个怀疑度计算公式计算每个变异体的怀疑度。将待测程序各语句块的怀疑度值设置为相关联的所有变异体的怀疑度的最大值,最后得到从大到小排序后的各语句块的怀疑度值表,推荐怀疑度高的语句块进行后续的排错工作。

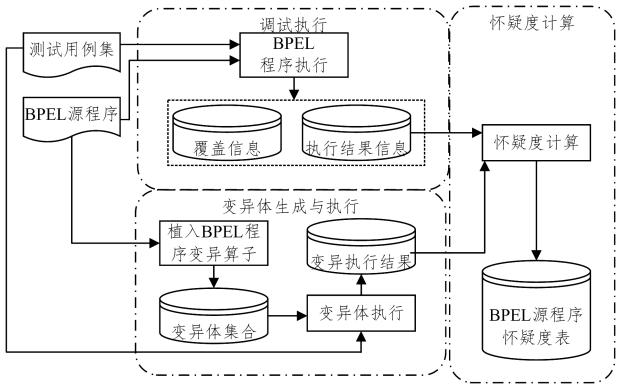

图2 BPEL程序故障定位方法框架

Fig. 2 Framework of fault location method for BPEL program

需要指出的是,对于未植入故障的语句块,将其怀疑度设为0。此外,当存在两个及以上的语句块怀疑度一样时,检查顺序将直接影响程序调试的效率。在BPEL程序中,如果一个具有故障的原子活动/结构化活动包含于另外一个结构化活动,则该活动的故障导致外层结构化活动也表现出故障的特征,因此应该优先检查外层的活动。如果不存在包含的活动,由于故障通常沿着执行路径进行传播,因此应该优先检查靠前的语句块。当语句块怀疑度一样时,采用如下启发式规则确定相同怀疑度语句块的故障定位次序:

规则1 在存在包含关系的活动中,优先检查外层的结构化活动,再检查被包含的活动;

规则2 在不存在包含关系的活动中,按照语句块在程序中出现的顺序依次进行检查。

3.2 定位算法

基于变异分析的故障定位算法的描述如算法1所示。该

算法的输入为 BPEL 源程序 P、测试用例集 T 和预期输出 E，输出为怀疑度排序后的语句块集合 S。其中，M 为生成的变异体集合；RT 为给定测试用例的预期结果 E 与源程序的执行结果 R 的比较结果的标记集合；RS 记录了给定变异体的执行轨迹；通过比较变异体的执行结果 RM 与源程序的执行结果 R，来确定给定变异体是否被杀死，相关信息记录在 RRT 中。算法 1 的主要过程如下：初始化通过测试用例的数量 $totalpassed$ 和未通过测试用例的数量 $totalfailed$ 为 0，初始化可疑集合、变异体集合、测试用例标记集合等为空集，见步骤 1；获取待测程序 P 的所有变异体，见步骤 2；对待测程序执行测试用例集，记录每个测试用例 t_i 的执行结果 $R(t_i)$，当 $R(t_i)$ 为 true 时，表示测试通过，否则表示未通过，将执行结果与预期结果进行对比，得到 RT 以及 $totalpassed$ 与 $totalfailed$，见步骤 3；计算每一个语句块的怀疑度，将语句块对应的变异体怀疑度最大值记为该语句块的怀疑度，具体见步骤 4—步骤 17；将语句块怀疑度由大到小排序，将排名靠前的语句块加入可疑集合 S，见步骤 18。其中，步骤 16 利用怀疑度计算公式计算语句块的怀疑度。

算法 1 基于变异分析的故障定位算法

INPUT：P={b_1，b_2，…，b_m}，T={t_1，t_2，…，t_n}，E={e_1，e_2，…，e_n}

OUTPUT：S={s_1，s_2，…，s_k}

PROCEDURE：

1. Initialize totalpassed and totalfailed as 0，and initialize RS，M，R，RT，RM，RRT，S as an empty set Φ
2. M ← Mutation Operation(P)
3. Execute P using T，get results R，compare R with E to get RT and calculate totalpassed and totalfailed
4. FOR each m_{is} in M DO
5. FOR each m_i in m_{is} DO
6. Initialize passed_killed and failed_killed as 0
7. Execute m_i using T and get results RM and RS
8. Compare RM with R and use RS to get RRT
9. IF (rrt(t_i) == '1')
10. passed_killed add 1
11. ELSE IF (rrt(t_i) == '0')
12. failed_killed add 1
13. END IF
14. susp (m_i) ← formula(Method，passed_killed，failed_killed，totalpassed，totalfailed)
15. END FOR
16. s_i = susp (b_i(m_{is}↔b_i)) ← Max(suspiciousness(m_i))
17. END FOR
18. S ← rank(s_i) and select Top(k) of amount(S) in front ranked
19. RETURN S

END PROCEDURE

3.3 方法示例

本节采用 BPEL 程序实例 QuoteProcess 示例本文提出的故障定位方法。QuoteProcess 模拟汽车维修用户的报价过程，调用了 6 个外部 Web 服务，其 BPEL 流程如图 3 所示。依据“VP_EstimateFunction”的表达式的值，选择不同的评估方案，当其为假时，同时进行普通评估“VP_interior”和专家评估“VP_powertrain”，评估结果通过“replyOutput”返回。

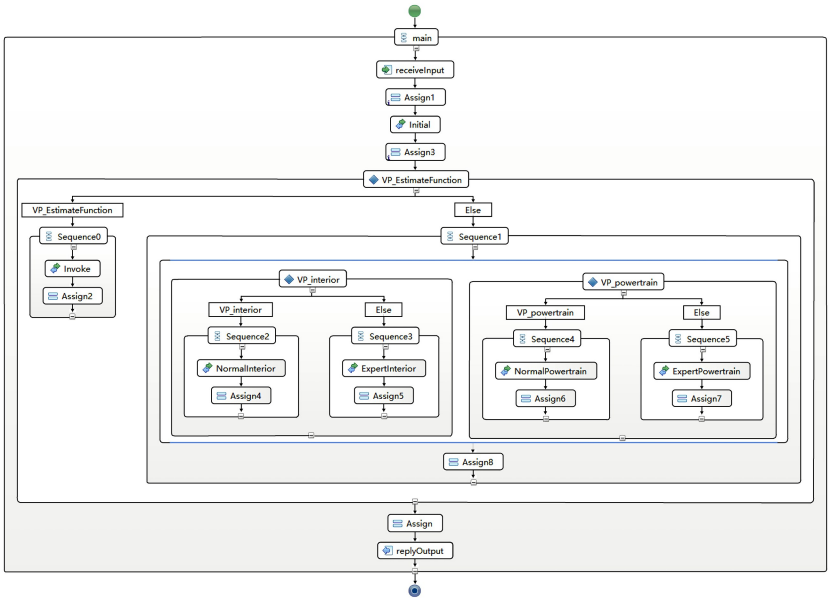

图 3 QuoteProcess 的 BPEL 流程图

Fig. 3 BPEL flow-process diagram of QuoteProcess

现假设 QuoteProcess 程序中存在一个 ERR 类型的故障，即“VP_interior”的条件表达式“$input.payload/tns:Amount<80”错误地实现为“$input.payload/tns:Amount>80”；采用 Tarantula 公式[11]计算怀疑度。本文方法定位上述故障的过程如下：

(1)解析 QuoteProcess 程序，获得该程序的语句块结构信息。

(2)对 QuoteProcess 程序的每一个语句块进行变异，得到相应的变异体。为了演示具有定位未知故障的能力，排除 ERR 类型的变异算子。

(3)对 BPEL 原始程序执行测试用例集，得到实际输出。将实际输出与期望输出进行对比，如果相同，则对应的测试用例 t 标记为“passed”；否则记为“failed”。

(4)对所有变异体执行测试用例集，将执行结果与原始程

序的执行结果进行对比。如果两者的结果相同，则该变异体被测试用例杀死；否则没有被杀死。统计每一个变异体被通过的测试用例杀死的数量和未通过的测试用例杀死的数量。

(5)采用 Tarantula 计算变异体的怀疑度，然后将语句块对应的变异体的怀疑度最大值作为该语句块的怀疑度。

(6)按照规则 1 与规则 2，对怀疑度相同的语句块进行重新排序，得到故障定位结果，具体如表 1 所列。

表 1 语句块怀疑度的排序结果

Table 1 Result of sorting statement block scepticism

Rank	Blocks	Rank	Blocks
1	VP_EstimateFunction	17	Assign3
2	**VP_interior**	18	Sequence0
3	NormalInterior	19	Sequence1
4	Assign4	20	Sequence3
5	Flow	21	Assign
6	Sequence2	22	replyOutput
7	VP_powertrain	23	ExpertInterior
8	Sequence4	24	Assign5
9	Sequence5	25	Invoke
10	ExpertPowertrain	26	Assign2
11	Assign7	27	else0
12	Assign8	28	NormalPowertrain
13	main	29	else1
14	receiveInput	30	Assign6
15	Assign1	31	else2
16	Initial	—	—

按照表 1 中推荐的语句块次序依次检查程序，直到找到真实的故障位置。本例中存在故障的活动是“VP_interior”，检查到第二个活动时就可以定位出故障位置。

3.4 优化策略

如前文所述，MBFL 技术应用于 BPEL 程序故障定位时，共有 34 种变异算子。如果这些变异算子全部用来生成变异体集合，将导致巨大的变异执行开销。结合 BPEL 程序特点与变异分析的优化技术，我们对所提故障定位技术进行优化，具体优化规则如下。

(1)变异算子选择优化：已有研究发现 BPEL 程序的变异算子中存在包含关系[8]，即 AEL←AIE，ASF←ASI，ERR←EIN，AEL←CFA，EIU←EAN，CDC←CDE，CDC←CCO。变异算子 A 包含 B，意味着若给定测试用例能杀死 A 产生的变异体，则也一定能杀死 B 产生的变异体。因此，在生成变异体时优先选择包含变异算子；包含变异算子已选用时，则排除被包含变异算子，从而减少生成的变异体数量，减小变异执行开销。

(2)基于语句块的细粒度变异体采样：分析 BPEL 程序的变异算子发现，“If”“While”等活动对应的语句块可以多次应用不同的变异算子。通过对不同变异算子产生的变异体集合进行采样(例如 10%，20%，50%)，不仅可以减少变异体的数量，同时还可以保留语句块对不同故障类型的敏感性。

本文提出的故障定位算法需要遍历所有的变异体，其时间复杂度为 $O(N)$，其中 N 为所有的变异体。优化规则 1 可以有效减少被包含变异算子生成的变异体，因此时间复杂度为 $O(N-M)$，其中 M 为被包含的变异算子生成的变异体数量。优化规则 2 则是通过选择部分变异体来减小算法开销，其时间复杂度为 $O(\alpha \cdot N)$，其中 α 表示采样率，且 $0<\alpha<1$。

4 支持工具

基于 Java 语言开发了面向 BPEL 程序的故障定位支持工具(DebugBPEL)。该工具的系统架构如图 4 所示，其主要构件的功能描述如下：

(1)BPEL 程序解析负责解析 BPEL 程序，获得 BPEL 程序的结构信息。

(2)程序执行负责 BPEL 程序的部署、测试用例消息的发送、返回结果的收集。执行对象包括待测的 BPEL 程序以及该待测程序生成的变异体集合。

(3)结果对比将待测程序的执行结果与期望输出进行对比。如果一致，则该测试用例记为通过，否则进行记为不通过。将变异体的执行结果与待测程序的执行结果进行对比，如果不一致，则该测试用例杀死变异体。

(4)怀疑度计算依据 BPEL 程序结构信息、待测程序和变异体的执行信息，采用不同的统计公式计算怀疑度值。

(5)结果输出对语句块怀疑度进行排序，将靠前的语句块作为定位结果推荐给用户。

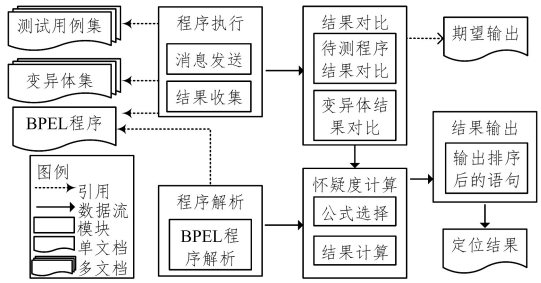

图 4 DebugBPEL 的系统架构

Fig. 4 System architecture of DebugBPEL

图 5 给出了 QuoteProcess 程序(见第 3.3 节中的讨论)的故障定位结果。为了便于调试人员应用故障定位结果，在 BPEL 源程序中标红推荐语句块，并给出推荐检查的次序。如上文的讨论，依据故障定位中的推荐结果，检查标红的第二个活动(“VP_interior”)时，便可定位到相应的故障。

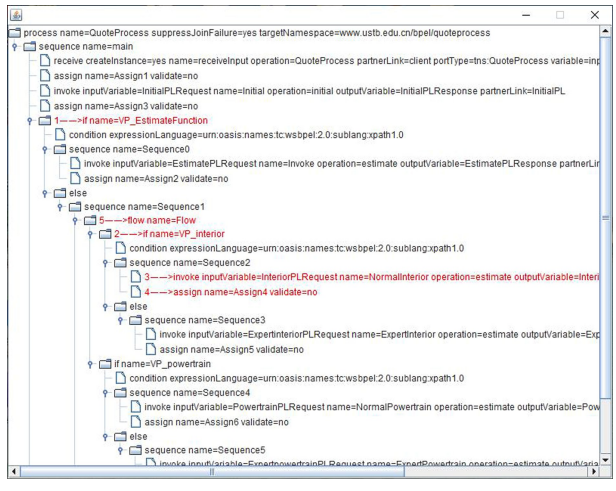

图 5 故障定位结果

Fig. 5 Fault location result

5 实验评估

本文采用 6 个 BPEL 程序作为研究对象(包括 SmartShelf[3]，QuoteProcess[12]，TravelAgency[12]，SupplyCustomer[3]，LoanApproval[12] 和 CarEstimate[12])，来验证提出的基于变异分析的 BPEL 程序故障定位技术的有效性。

5.1 度量指标

本文采用召回率、定位代价和变异测试对这 3 种度量指标，来评估故障定位方法的有效性，并与已有故障定位方法进行比较。

(1)召回率：给定故障集合总数 N，使用故障定位技术能够定位出的故障数 M，则故障召回率 Recall$=M/N$。显然，召回率越大，故障定位方法越有效。

(2)定位代价：给定一个 BPEL 故障程序，活动总数为 A，使用故障定位技术得到怀疑度表(怀疑度从大到小)，按顺序检查活动至定位出实际故障所需检查的活动数为 B，则故障定位代价 $Cost=B/A$。显然，定位代价数值越小，故障定位方法越高效。

(3)变异测试对：一次变异执行中的变异体与测试用例记为一个 MTP(Mutation Test Pair)。执行的 MTP 次数越多，执行开销就越大。

5.2 研究问题与实验设计

通过实验评估回答如下 3 个研究问题。

RQ1：基于变异分析的 BPEL 故障定位技术的有效性如何？

采用本文方法对 6 个代表性的 BPEL 程序生成的所有故障版本进行故障定位，得到每个故障版本的怀疑度表，从故障定位的召回率与精确度两个方面评估本文方法的有效性。

RQ2：本文提出的优化策略能否减少变异执行开销？

分别统计原始的 MBFL 技术与优化后的技术定位每一个故障版本所需的 MTP 执行次数，通过 MTP 减小的比例来评估优化方案的有效性。

RQ3：与已有的 BPEL 程序故障定位技术相比，本文方法的故障定位效果如何？

已有的 BPEL 程序故障定位技术包括基于块结构的故障定位方法[3]、基于谓词切换的方法(Switching)[4]、基于谓词切换与程序切片的方法(Slicing)[5]。基于块结构的故障定位方法采用 Tarantula 方法[11]计算怀疑度公式(Tarantula)；考虑到 DStar 方法[13]是最近提出的程序谱方法，因此将其合成到基于语句块的故障定位方法，简称为 DStar。本文提出的基于变异分析的故障定位方法，采用 Tarantula 方法与 DStar 方法计算怀疑度，分别记为 MBFL-T 和 MBFL-D。按照不同基线技术的定位结果，检查按怀疑度排序的语句块，得到不同方法对应的召回率和定位代价。

由于不同基线技术的适用条件不一致，本文的实验评估方案设计如下。1)测试用例集设置为 MBFL-T，MBFL-D，Tarantula 和 DStar，利用测试用例集的相关信息(包括每个测试用例的语句块覆盖、测试是否通过等)进行故障定位；Switching 与 Slicing 仅从测试用例集中随机挑选一个执行失败的测试用例，得到与故障相关的谓词和变量集合，检查可疑集合中的元素得到召回率和定位代价。2)实验程序的故障设置如下：本文采用变异分析模拟真实的 BPEL 程序故障，然后用这些故障评估不同故障定位方法的有效性。为了保证评估的公正性，在验证本文方法时采用留一验证法进行故障设置。具体说来，针对给定的 34 种 BPEL 程序变异算子，每次故障定位只使用一个变异算子生成变异体作为故障版本(即验证集)；在故障定位过程中，从生成的全部变异体集中移除该变异算子生成的变异体(即训练集)；采用上述方式，可以有效验证本文方法定位未知故障的能力。

5.3 实验结果分析

5.3.1 故障召回率评估结果与分析

本文采用本文方法与基线技术对 6 个实验程序进行了故障定位实验，不同方法的故障召回率实验结果比较如图 6 所示。召回率评估结果的总结与分析如下：

(1)本文方法(MBFL-T 和 MBFL-D)在所有的实验程序对象上，均表现出最好的定位效果，召回率最高达到 90%。不同怀疑度计算公式对本文方法的召回率的影响不明显。

(2)基于谓词切换与程序切片的方法 Slicing 在故障定位召回率方面的表现仅次于本文方法，但均优于其他方法。通过进一步分析发现，Slicing 与 Switching 无法定位活动顺序交换的故障类型，在活动顺序交换后不能记录执行轨迹，而本文方法只需要知道执行结果前后的差异，不需要执行轨迹信息。

(3)基于谓词切换的方法 Switching 与基于程序谱的方法 DStar 的召回率基本相当。其中，Switching 在实验程序 SmartShelf 与 CarEstimate 上的表现明显优于 DStar，在实验程序 QuoteProcess，SupplyCustomer，LoanApproval 上的表现略差于 DStar。

(4)基于程序谱的方法 Tarantula 的召回率在所有实验程序上表现最差(在 SmartShelf 与 CarEstimate 上 Tarantula 与 DStar 基本相当)。通过进一步分析发现，测试用例集影响基于程序频谱的方法。给定一个故障，当所有测试用例都执行不通过时，Tarantula 计算公式中的分母为 0，无法定位此类故障。

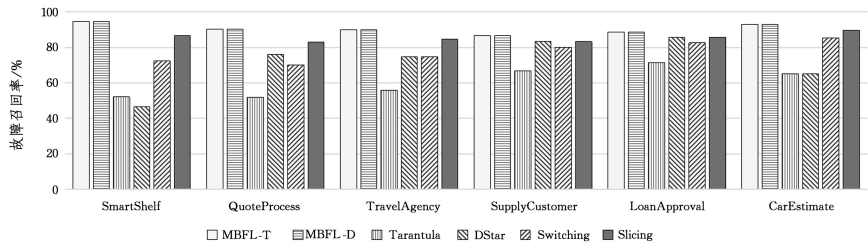

图 6 故障定位召回率的实验结果

Fig. 6 Experimental results of fault location recall rate

综上，本文方法在 6 个实验对象上均表现出了最好的定位结果，基于谓词切换与程序切片的方法次之，基于程序谱的方法最差。

5.3.2 故障定位代价评估结果与分析

图 7 给出了不同方法的故障定位代价实验结果。故障定位代价结果的总结与分析如下。

(1)将本文方法应用于 6 个实验程序得到的故障定位代价为 4%～15%。这意味着本文方法推荐的语句块在检查程序时，能较快地定位到故障的位置。MBFL-T 和 MBFL-D 的定位代价非常接近，这表明怀疑度计算公式对本文方法

的定位效果的影响不明显。

(2)基于谓词切换与程序切片的故障定位方法 Slicing 的定位代价最低,其次是本文方法 MBFL-T 和 MBFL-D,再次是基于程序频谱的方法 Tarantula 和 DStar,而基于谓词切换的方法 Switching 的故障定位代价最高。此外,Tarantula 比 DStar 的定位代价低。Slicing 的定位代价低的主要原因是,该方法在找到关键谓词后对关键谓词进行数据切片,因此能够更精确地定位故障的位置。

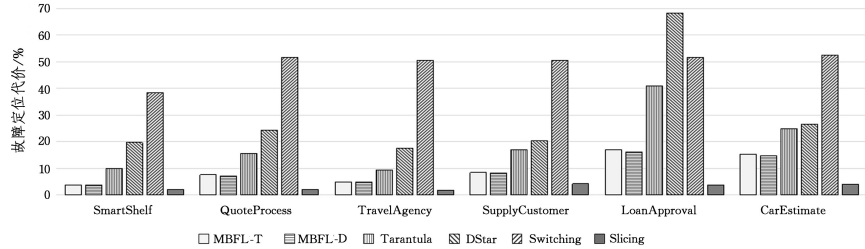

图 7 故障定位代价的实验结果

Fig. 7 Experimental results of fault location cost

5.3.3 变异执行开销评估结果与分析

图 8 给出了使用优化策略前后的本文方法的故障定位代价实验结果。其中,Pre-Su 表示未采用第 3.4 节提出的优化策略的故障定位效果;Aft-Su 表示采用了第 3.4 节提出的优化策略的故障定位效果,其中采样率为 20%。图 9 比较了本文方法优化前后的变异执行开销。

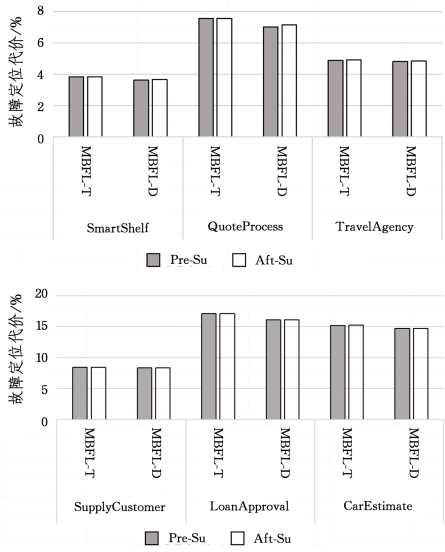

图 8 优化前后定位代价的实验结果

Fig. 8 Experimental results of fault location cost before and after optimization

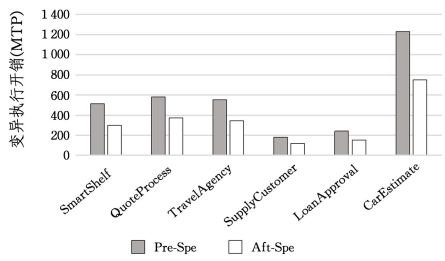

图 9 优化前后的变异执行开销

Fig. 9 Mutation execution cost before and after optimization

从图 8 和图 9 可以看出,本文提出的优化策略可以明显减小变异执行开销,但并不显著影响故障定位代价。

上述实验结果表明,与已有的 BPEL 程序故障定位技术相比,本文方法具有更好的故障定位召回率,但故障定位代价略高于谓词切换与程序切片相结合的方法;此外,本文提出的优化策略可以有效降低变异执行开销且不影响故障定位的代价。基于上述评估,本文方法提供了一种 BPEL 程序故障定位的新途径。

6 相关工作

如何保证计算机软件的质量一直是软件开发的核心问题之一。随着软件的规模与复杂性急剧增加,传统手工的故障定位方式耗时费力。近年来,人们开始探索自动化的故障定位技术,提出了基于切片、基于频谱、基于统计、基于程序状态、基于机器学习、基于数据挖掘和基于模型等多种故障定位新技术[1]。下文介绍基于变异分析的故障定位技术和面向 BPEL 程序的故障定位技术,并与本文工作进行比较。

基于变异分析的故障定位技术 MBFL 属于基于统计的故障定位技术范畴。MBFL 的主要思想[6]是:对待测程序进行变异处理生成变异体,通过对比源程序与变异体的测试结果来计算变异体怀疑度,将变异体怀疑度的最大值作为该语句的怀疑度,以此定位出故障位置。Papadakis 等开发出 MBFL 支持工具 Metallaxis-FL[14]。He 等利用变异分析修正故障定位结果[15],降低了偶然性成功测试用例对故障定位的影响,提高了定位的召回率。Moon 等[16]依据对错误语句进行变异可能使更多的测试用例通过、对正确的语句进行变异可能使更多的测试用例不通过这一观测,进一步改进了 MBFL 技术。虽然 MBFL 的定位精度较高,但是执行开销大,选择变异算子、变异体采样、优化变异执行等有助于减小 MBFL 的执行开销。Papadakis 等通过选择变异算子减少生成变异体的数量[17],当生成的变异体规模减少 80%时,MBFL 具有相同的故障定位效果。Liu 等通过变异体采样减少变异体数量[18],在语句级别选择一定比例的变异体的同时考虑变异体的多样性,在损失故障定位精度的同时减小变异执行开销。Wang 等提出面向语句的变异体约减策略[19],通过分析测试用例的执行信息,来对失败测试用例覆盖的语句生成的变异体集合进行约简,在保持较高故障定位精度的同时减少 70%以上的变异执行开销。Gong 等提出一种动态变异约减策略[20],通过搜集测试用例执行信息来动态地调整变异体及测试用例的执行顺序,避免执行不必要的测试用例,以减小 MBFL 的变异执行开销。本文探讨了面向 BPEL 程序的 MBFL 技术,针对 BPEL 程序的特点提出了有效的 MBFL 故障定位框架与优化策略。

近年来,研究人员从不同角度探索了新型 BPEL 程序的

故障定位问题。Sun 等较早研究了 BPEL 程序的故障定位问题,提出一种基于语句块的 BPEL 程序定位框架[3]。该框架将 BPEL 程序划分为多个语句块,利用测试用例的执行信息计算每个语句块的怀疑度,同时结合 BPEL 程序的特点提供了检查语句块的启发式规则。该框架合成多种基于频谱的怀疑度计算公式,实验结果表明合成的故障定位技术的召回率约为 50%。基于谓词切换的 BPEL 程序故障定位技术通过谓词切换缩小故障搜索范围[4],从关键谓词入手分析故障的根源,该技术的故障定位精度存在一定的不足。Sun 等进一步将谓词切换与程序切片相结合来探索 BPEL 程序的故障定位问题,提出了谓词切换与程序切片相结合的 BPEL 程序故障定位技术[5],其首先通过谓词切换找到关键谓词,然后对关键谓词进行后向数据切片。该方法的故障定位效果好,具有较高的故障定位召回率和较低的定位代价,但是仍无法定位出变量声明和活动交换等故障类型。针对上述问题,本文从变异分析角度出发,探索更有效的 BPEL 程序故障定位方法,提出了一种面向 BPEL 程序的故障定位框架。该框架借鉴了基于语句块的 BPEL 故障定位思想[3]和相应的优化策略,集成了 BPEL 程序变异测试优化方面的成果[8-9]。实验结果表明,本文方法在提高 BPEL 程序故障定位召回率的同时,有效降低了变异执行开销。

结束语 本文提出了一种基于变异分析的 BPEL 程序故障定位技术,开发了相应的支持工具。所提技术将变异分析的思想应用于 BPEL 程序的故障定位中,并根据 BPEL 程序和其变异算子的特点对所提故障技术进行优化。采用 6 个 BPEL 程序实例对本文提出的方法进行评估,实验结果表明:1)与 Tarantula 和 DStar 等基于程序谱的故障定位技术相比,本文方法具有较高的召回率和较低的定位代价;2)与谓词切换和程序切片相结合的故障定位技术相比,本文方法具有较高的召回率,但定位代价略高;3)本文提出的优化策略可以在保持故障定位代价的情况下减少变异执行开销。

未来将采用更多 BPEL 程序实例来验证本文方法的有效性;此外,在变异分析研究进展的基础上,研究如何进一步降低本文方法的变异执行开销。

参 考 文 献

[1] WONG W E,GAO R,LI Y,et al. A survey on software fault localization [J]. IEEE Transactions on Software Engineering, 2016,42(8):707-740.

[2] SUN C A. Some Open Issues in SOA software development [EB/OL]. Beijing: Science Paper Online. http://www.paper.edu.cn/releasepaper/content/201107-461.

[3] SUN C A,ZHAI Y M,SHANG Y,et al. BPELDebuger: An effective BPEL-specific fault localization framework [J]. Information and Software Technology,2013,55(12):2140-2153.

[4] ZHENG C Y. A Predicate Switching based Approach to Locating Faults of BPEL Programs and Supporting tool[D]. Beijing: University of Science and Technology Beijing,2015.

[5] SUN C A,RAN Y F,ZHENG C Y,et al. Fault localization for WS-BPEL program based on predicate switching and program slicing[J]. Journal of Systems and Software,2018,135(1):191-204.

[6] PAPADAKIS M,TRAON Y L. Using mutants to locate "Unknown" faults[C]// Proceedings of the fifth International Conference on Software Testing. Verification and Validation,IEEE Computer Society,2012:691-700.

[7] 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,2018,11(4):616-629.

[8] SUN C A,PAN L,WANG Q L,et al. An empirical study on mutation testing of WS-BPEL programs [J]. The Computer Journal,2017,60(1):143-158.

[9] SUN C A,WANG Z,PAN L. Optimized Mutation Testing Techniques for WS-BPEL Programs[J]. Journal of Computer Research and Development,2019,56(4):895-905.

[10] ESTERO-BOTARO A,MEDINA-BULO I. MuBPEL [EB/OL]. https:// neptuno. uca. es/redmine/projects/sources-fm/wiki/MuBPEL. html.

[11] JONES J A,HARROLD M J,STASKO J. Visualization of test information to assist fault localization[C]// Proceeding of the 24th International Conference on Software Engineering. IEEE Computer Society,2002:467-477.

[12] Eviware. Web services business process execution language [EB/OL]. http://docs. oasis-open. org/wsbpel/2. 0/OS/wsbpel-v2. 0-OS. html.

[13] WONG W E,DEBROY V,GAO R,et al. The DStar method for effective software fault localization [J]. IEEE Transactions on Reliability,2014,63(1):290-308.

[14] PAPADAKIS M,TRAON Y L. Metallaxis-FL: mutation-based fault localization [J]. Software Testing,Verification and Reliability,2015,25(5/6/7):605-628.

[15] HE T,WANG X M,ZHOU X C,et al. A Software Fault Localization Technique Based on Program Mutations [J]. Chinese Journal of Computers,2013,36(11):2236-2244.

[16] MOON S,KIM Y,KIM M,et al. Ask the mutants: mutating faulty programs for fault localization[C]// Proceedings of the 2014 IEEE Seventh International Conference on Software Testing,Verification and Validation. IEEE Computer Society,2014:153-162.

[17] PAPADAKIS M,TRAON Y L. Effective fault localization via mutation analysis: a selective mutation approach[C]// Proceedings of the 29th Annual ACM Symposium on Applied Computing. ACM press,2014:1293-1300.

[18] LIU Y,LI Z,WANG L X,et al. Statement-oriented mutant reduction strategy for mutation based fault localization[C]// Proceedings of the 2017 IEEE International Conference on Software Quality,Reliability and Security. IEEE Computer Society,2017:126-137.

[19] WANG L X,WANG W W,ZHAO R L,et al. MBFL with Statement-oriented Mutant Reduction Strategy[J]. Computer Science,2017,44(11):175-180.

[20] GONG P,GENG C Y,GUO J X,et al. Dynamic Mutation Execution Strategy for Mutation-based Fault Location [J]. Computer Science,2016,43(2):199-203,229.

SUN Chang-ai, born in 1974, Ph.D, professor, Ph. D supervisor, is a senior member of China Computer Federation. His main research interests include software testing, program analysis and service oriented computing.