目录

1. 背景介绍

计算机软件正广泛应用于经济、军事、商业等各个领域中，对其可靠性问题的研究也日益得到了人们的广泛重视。软件中的故障通常由程序故障引起，程序故障是指程序在运行过程中出现的一种不正常的内部状态。故障定位是确定程序中可疑故障的确切性质和位置，为故障修复做准备。程序故障定位的有效性受到程序规模、程序设计语言和开发人员能力等因素的影响，导致程序故障定位成为调试过程中一项耗时费力的工作。为了降低开发维护成本，帮助软件测试人员找到软件故障，研究自动化故障定位技术具有十分重要的现实意义。

近年来，面向服务架构(Service-Oriented Architecture)逐渐成为互联网环境下应用程序开发的首选解决方案。Web 服务作为开发 SOA 应用的基本单元，构建在开放的 XML 标准和 Web 协议上，具有平台独立、松耦合的特点。由于单个的 Web 服务往往实现的功能有限，无法满足实际业务需求，为了构建复杂的业务流程需要将多个 Web 服务协调并组织起来构造新的 Web 服务（复合服务。WS-BPEL 是一种基于 XML 的服务组装语言，广泛应用于开发基于 Web 服务的分布式程序。与传统程序相比，WS-BPEL 程序存在如下的新特性：

a) WS-BPEL 程序以 XML 文件的形式呈现[9]，与传统程序的语法结构有着很大的区别。

b) WS-BPEL 程序提供了一个明显的组合机制，可以以伙伴连接的形式调用其他外部服务，并且外部服务可能是由不同的语言实现。

c) WS-BPEL 程序以活动（Activity）为单位，定义了一套完整的执行语句，如顺序、选择、并发等控制结构，所有的控制结构都以 XML 的格式存在。

d) WS-BPEL 提供了用流程活动来表示同步，用流程中的链接来表示并发，这种表达形式在一般程序中较为罕见。

上述特点使得 WS-BPEL 程序的测试与调试面临新的挑战。目前在 WS-BPEL 故障定位方面已经取得一些研究成果，包括合成的 Tarantula、基于谓词切换和基于谓词切换与程序切片的 WS-BPEL 程序故障定位技术。

1.1 面向服务的架构（SOA）

面向服务架构是一种构建分布式系统的架构，支持程序实现的功能以服务的方式对外发布，每个服务既可以独立的发布也可以组合为更复杂的服务对外发布。SOA 的模型如图 1-1 所示。

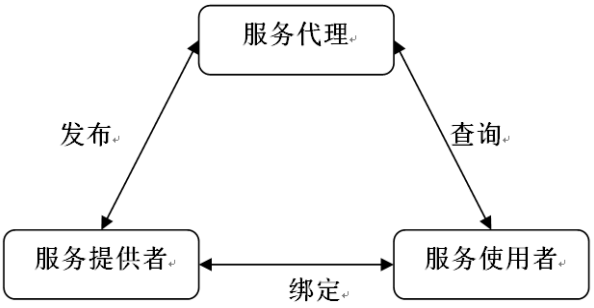

图 1-1 SOA 模型

该模型中存在协同合作的三种角色，即服务提供者、服务使用者和服务代理。其中，服务提供者将开发成功的服务以某种方式挂载在服务代理上面，供服务使用者检测和使用该服务；服务代理使用服务库的方式存放各式各样的服务，供服务使用者从中选择所需的服务，起到连接提供者和使用者的作用；服务使用者在需要服务时，会到服务代理库中寻找所需的服务，找到后按照某种协议使用该服务。

1.2服务组装语言

Web 服务一般遵循以下几个协议标准：简单对象访问协议 SOAP 是定义访问 Web 服务的协议；Web 服务描述语言 WSDL 是描述 SOAP 协议的具体语言；统一描述、发现和整合规范 UDDI，将 Web 服务收集和存储起来，当用户需要访问某个服务时就从 UDDI 里面查找。Web 服务实现了对业务逻辑的封装，并以独立于操作系统和开发平台的方式向外提供调用接口。

在 SOA 模型各层次中企业已有的程序资源首先被封装为业务组件，继而使用这些组件或它们的复合建立起具有标准接口的可重用服务。服务通过标准的 WS-BPEL 语言进行编排形成业务流程层，并最终在企业信息门户等平台中进行集中展示。WS-BPEL 是进行 Web 服务编排和协调的首选标准流程语言，支持服务操作之间常见的时序与逻辑关系，还可以表示业务伙伴之间的控制依赖关系。为了提供可靠的业务流程的执行，BEPL 支持事务与补偿机制和异常管理，确保流程执行的数据一致性。

WS-BPEL 流程将过程组件通过 WS-BPEL 活动连接起来。常用的组件有伙伴连接、变量、相关集、错误处理和补偿处理。

a) **伙伴连接**：WS-BPEL 中的服务提供者，主要分为两种，一种是 WS-BPEL 流程所要调用的外部服务，另一种是指 WS-BPEL 自己所要提供的服务。

b) **变量**：WS-BPEL 需要集成 WSDL 定义的 Web 服务，在 Web 服务之间的交互过程中，变量用来存储 Web 服务的输入与输出。

c) **相关集**：用一组特定的数据，来关联和标定一个 WS-BPEL 过程实例。

d) **错误处理**：WS-BPEL 程序在调用合作伙伴的过程中，或者合作伙伴的服务有可能会抛出异常，或者 WS-BPEL 流程内部调用中也会抛出异常。WS-BPEL 流程中有一套机制可以捕获异常，并可以定义捕获异常后的下一步行动。

e) **补偿处理**：WS-BPEL 流程需要处理的一系列业务流程中，如果一个活动出现异常之后，有些业务流程需要将取消它前面的活动。

WS-BPEL 活动是指 WS-BPEL 流程中一个条语句或者一个步骤的执行，分为基本活动和结构化活动。基本活动在程序中被称为原子活动，结构化活动在程序中被称为非原子活动。基本活动包括：

Receive：等待接收外部客户端或伙伴服务传来的消息，从而使流程完成一定的功能。

Invoke：调用某个伙伴服务，可以是同步的请求响应操作，也可以是单向的异步操作。

Reply：将输出结果返回客户端。

Wait：等待一定的时间和时间段。

Assign：更改流程内部状态，将值从源变量赋值到目标变量。

Throw：抛出异常。

Exit：终止整个流程实例

Empty：空活动，没有实际意义，但是可以用来同步活动。

Compensate：以默认顺序开始恢复那些已成功完成的所有内部域中的活动。

CompensateScope：用来恢复已成用完成的某一特定域内的活动。

结构化活动用来约束其孩子活动之间的执行顺序，结构化活动的各个活动表示的意义如下所示：

Sequence：按照活动定义的先后顺序依次执行，在最后一个活动完成后，Sequence 活动才完成。

If：按照一定的条件执行某一特定的操作。

Pick：基于消息的事件选择机制，当匹配的消息到达时，执行特定的消息事件。

While，RepeatUntil：循环执行某个活动，前者先判断后执行，后者先执行后判断执行条件是否满足。

Flow：执行活动间的可以并发执行，通过 link 结构表达并发活动之间的同步控制依赖，只有当其包含的子活动全部执行完毕(或被忽略执行)，flow 活动才完

成。

Scope：执行活动执行的作用域，它可以关联事件处理(eventHandlers)、错误处理(faultHandlers)以及补偿处理(compensationHandier)。

其中“receive”、“invoke”、“reply”和“pick”是用来建模 BPBL 流程通信行为的交互活动。此外，“exit”也能影响流程的通信，当执行“exit”活动后，正在执行的整个流程实例将终止执行。“assign”赋值活动也是一个重要活动，可以用来改变流程的执行状态。“empty”活动尽管不具有任何语义，但可以用来不同并发活动之间的控制依赖关系。

1.3 WS-BPEL 故障类型

在变异测试中，对被测试程序的源代码植入各种合乎语法的故障，形成一个包含故障的故障版本，植入的故障类型称为变异算子。已经提出的 WS-BPEL 语言的故障类型可以分为四大类，包括标识符替换、活动替换、表达式替换和异常与事件类型替换，活动替换又分为并发相关和并发无关。Estero-Botaro 等人针对 WS-BPEL 程序提出了 26 种的变异算子，通过这些变异算子来指导产生变异体，变异体作为测试数据来评估面向 WS-BPEL 程序测试技术的故障检测能力。Garc á-Dom ńguez 等人更新了 WS-BPEL 变异算子种类，新增 *EIN*，*EIU*，*EAP* 和 *EAN* 四个变异算子到表达式变异算子类型中，并新添覆盖准则变异算子类型，包括 *CFA*，*CDE*，*CDC* 和 *CCO*，如表 1-1 所示。

表 1-1 WS-BPEL 的变异算子描述

变异算子	描述
标识符替换变异算子	
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	
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(<， >，<=， >=， =， !=)用另一相同类型的运算符替换
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

表 1-2 WS-BPEL 的变异算子描述（续）

变异算子	描述
EMF	将截止时间用 0 或者是其一半代替
EIN	插入 XPath 格式 not 到逻辑表达式中
EIU	插入 XPath 格式一元减法到算数表达式中
EAP	用表达式的绝对值替代该表达式
EAN	用表达式的绝对值的负值替代该表达式
并发相关活动的变异算子	
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”
并发无关活动的变异算子	
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	
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
覆盖准则变异算子	
CFA	用 exit 活动替代其他活动
CDE	应用判断覆盖准则到程序中
CCO	应用条件覆盖准则到程序中
CDC	应用判断/条件覆盖准则到程序中

课题组前期提出了基于谓词切换的 WS-BPEL 故障定位技术，通过执行能发现故障的测试用例来记录程序的执行轨迹，找出包含故障的谓词。通过进一步研究，课题组将谓词切换与程序切片两种故障定位技术结合起来，提出了基于谓词切换与程序切片的 WS-BPEL 故障定位技术，相关的成果以 Fault localization for WS-BPEL Program Based on Predicate Switching and Program Sclicing 为题发表在 Journal of System and Software。

2. WS-BPEL_CSLocator 系统的分析与设计

2.1 需求分析

为了实现针对 WS-BPEL 程序故障定位的自动化，以及更加快捷的使用本文提出的故障定位技术，开发了一个基于控制依赖和统计分析的 WS-BPEL 程序故障定位工具 WS-BPEL_CSLocator。该工具的主要功能包括文件导入、程序解析、测试用例执行、可疑集合缩减、谓词排序和定位结果输出。对 WS-BPEL_CSLocator 采用 UML 用例图进行需求分析，如图 2-1 所示。

(1) **导入文件**：该用例包含 WS-BPEL 文件、测试用例集和预期输出的导入三个部分。
- WS-BPEL 文件：用户提供的待测 WS-BEPL 程序。
- 测试用例集：用户提供的能发现该 WS-BPEL 程序故障的测试用例集。
- 预期输出：用户提供的预期输出结果文件。

(2) **程序解析**：该用例包括解析 WS-BPEL 程序。
- 解析 WS-BPEL：该用例解析用户导入的 WS-BPEL 程序，获得 WS-BPEL 程序活动的名称和活动间的控制依赖信息。

(3) **测试用例执行**：该用例主要包括发送消息和收集轨迹。
- 发送消息：该用例向 WS-BPEL 程序发送用户提供的测试用例并返回程序执行结果。
- 收集轨迹：该用例将当前测试用例的执行结果和期望输出结果对比，并收集执行测试用例后的执行轨迹。

(4) **可疑集合缩减**：该用例依据收集的执行轨迹集以及执行结果，将执行轨迹集分为失败执行轨迹集 *FRS* 和成功执行轨迹集 *PRS*，通过活动间的控制依赖关系，减少可疑集合中活动的数量。
- 判断谓词分支：在 *FRS* 中，当可疑集合中的谓词的真分支和假分支支数量都大于 1 时，根据程序活动间的控制依赖关系，在可疑集合中删除控制依赖于该谓词的活动。
- 判断公共原子活动：当 *PRS* 中执行轨迹的数量大于 0 是，在可疑集合中删除公共原子活动。

(5) **谓词排序**：该用例通过统计可疑集合中谓词表达式在失败执行轨迹 *FRS* 和成功执行轨迹 *PRS* 中取真值的次数，利用 SOBER 故障定位技术对可疑集合中的谓词进行怀疑度计算并排序。

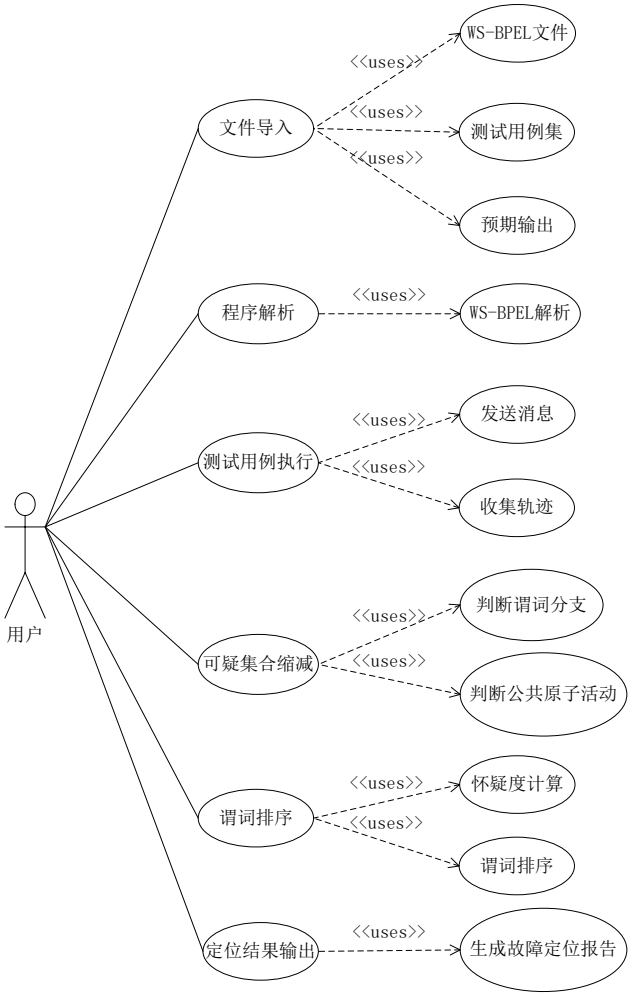

图 2-1 WS-BPEL_CSLocator 系统用例图

- 怀疑度计算：统计可疑集合中的谓词在 *FRS* 和 *PRS* 中表达式取真值的次数，根据 SOBER 故障定位技术中的公式，计算谓词怀疑度。
- 谓词排序：根据计算出的谓词怀疑度对可疑集合中的谓词由高到低进行排序。

(6) **定位结果输出：**该用例根据可疑集合中的谓词排序结果和可疑集合生成故障定位结果报告。

生成故障定位报告：根据可疑集合中排序后的谓词和可疑集合，先输出按顺序输出可疑集合中的谓词，再输出可疑集合中的非谓词节点。

2.2 系统设计

图 2-2 描述了工具 WS-BPEL_CSLocator 的系统架构，矩形框表示工具的基本组件，包括 WS-BPEL 程序解析、测试用例执行、可疑集合缩减、谓词排序和

定位结果输出 5 个组件，各个组件的功能设计如下：

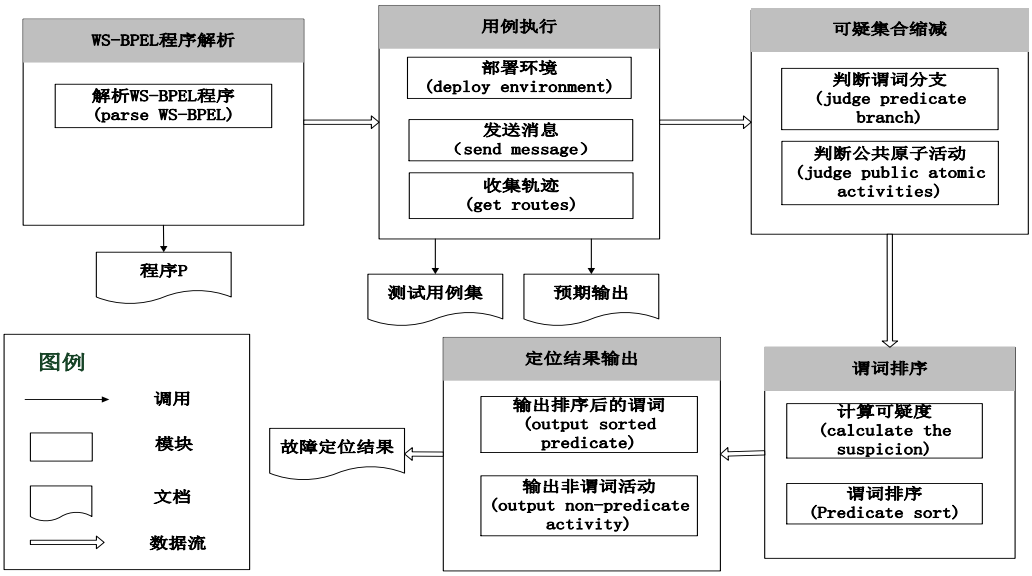

图 2- 2　WS-BPEL_CSLocator 系统架构图

“WS-BPEL_CSLocator”的功能模块包括 WS-BPEL 程序解析、测试用例执行、可疑集合缩减、谓词排序和定位结果输出五个部分。

(1) WS-BPEL 程序解析模块中主要对 WS-BPEL 程序静态解析，解析每个活动的名称以及活动间的控制依赖关系。

(2) 测试用例执行模块包括部署 WS-BPEL 程序执行环境、发送测试用例、获取执行轨迹功能。Apache ODE 引擎可以通过添加新的 WS-BPEL 文件自动部署服务，“AXIS2”包对输入的测试用例进行“SOAP”消息格式封装。测试用例结束后，WS-BPEL 流程将输出结果以“SOAP”格式返回，记录返回的执行结果并和预期结果进行比较，并记录执行轨迹。

(3) 可疑集合缩减模块包括判断谓词分支和判断公共原子活动是否存在故障的功能。收集的到执行轨迹集根据实际输出和预期输出的比较结果分为失败执行轨迹集 *FRS* 和成功执行轨迹集 *PRS*，将失败执行轨迹集 *FRS* 中的活动都加入可疑集合中。在 *FRS* 中，可疑集合中的任一谓词的真分支和假分支都是多个时，判定控制依赖于这个谓词的结点不存在故障，把控制依赖于该谓词的活动从可疑集合中删除。在 *PRS* 中的轨迹是多个时，判定公共原子活动一定不存在故障，把公共原子活动从可疑集合中删除。

(4) 谓词排序模块需要分别统计可疑集合中的谓词在失败执行轨迹 FRS 和成功执行轨迹 PRS 中条件表达式取真值的次数，根据 SOBER 故障定位技术中的公式对谓词进行怀疑度计算，并按照怀疑度由高到低对谓词进行排序。

(5) 定位结果输出模块根据谓词排序结果输出可疑集合中的谓词，再输出可疑集合中的非谓词结点，得到故障定位结果报告。

3. WS-BPEL_CSLocator 系统的实现与演示

3.1 系统实现

本小节介绍支持工具 WS-BPEL_CSLocator 的实现。WS-BPEL_CSLocator 核心功能的类包括程序解析类、测试用例执行类、可疑集合缩减类、谓词排序类和定位结果输出类，WS-BPEL_CSLocator 主要的核心类图如图 3-1 所示，下面介绍这几个类。

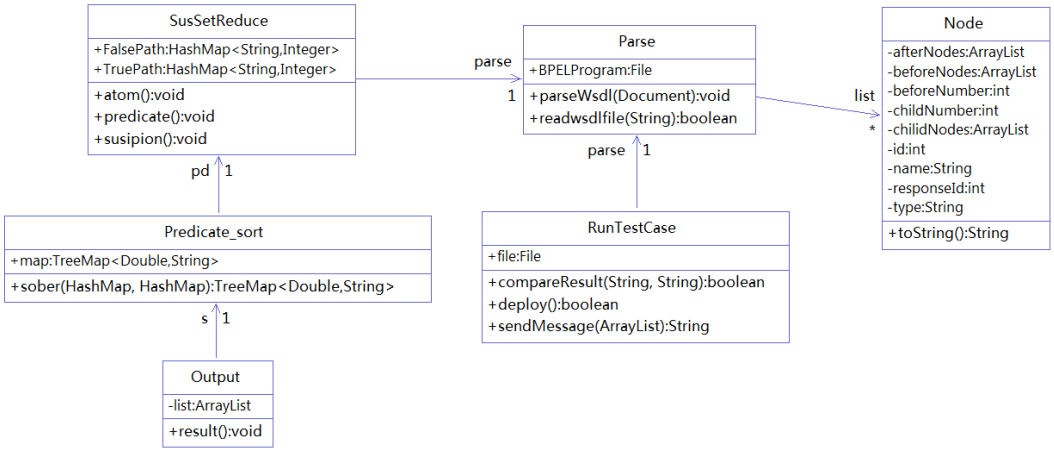

图 3- 1 WS-BPEL_CSLocator 工具类图

(1) WS-BPEL 程序解析 Parse 类：使用 DOM4J 技术对输入的 WS-BPEL 程序进行解析，并存储程序中活动的名称以及活动间控制依赖关系等信息。

(2) 测试用例执行 RunTestCase 类：用于配置 WS-BPEL 程序执行环境并执行测试用例集，收集测试用例执行结果和执行轨迹。

Deploy()：该方法用于配置 WS-BPEL 程序执行环境并部署服务。

SendMessage()：把测试用例集保存在 ArrayList 数据结构中，依次向 WS-BPEL 程序发送“SOAP”格式的测试用例，执行测试用例并记录执行轨迹。

CompareResult()：根据测试用例执行结果和预期输出比较，记录当前测试用例是失效测试用例还是成功测试用例。

(3) 可疑集合缩减 SusSetReduce 类：用于得到可疑集合，并利用执行轨迹和 WS-BPEL 程序活动间的控制依赖关系缩小可疑集合。

Suspicion()：把失败执行轨迹集 *FRS* 中的结点都加入到可疑集合中。

Predicate()：可疑集合中的谓词在 *FRS* 中的真分支和假分支出现次数如果都大于 1，根据解析得到的程序活动间的控制依赖关系，从可疑集合中删除控制依赖于该谓词的结点。

Atom()：如果 *PRS* 中执行轨迹的数量大于 0，把公共原子活动从可疑集合中删除。

(4) 谓词排序 Predicate_sort 类：对可疑集合中的谓词进行怀疑度计算，并根据怀疑度从高到低对谓词进行计算。

Sober()：统计可疑集合中的谓词在失败执行轨迹集 *FRS* 和成功执行轨迹集 *PRS* 中取真值的次数，对该谓词进行 SOBER 算法的怀疑度计算，并对可疑集合中的谓词根据怀疑度由高到低进行排序。

(5) 定位结果输出 Output 类：生成故障定位报告。

Result()：先输出排序后的可疑集合中的谓词，再输出可疑集合中的非谓词结点。

3.2 系统演示

故障定位工具 WS-BPEL_CSLocator 以“QuoteProcess”程序实例进行演示，包括文件导入、测试用例执行、可疑集合缩减、谓词排序和定位结果输出部分。

第一步：图 3-2 所示的文件选择界面包括三个文件选择按钮，点击“OPEN WS-BPEL”，选择待测的 WS-BPEL 程序；点击“Choose Testcases”，选择需要执行的测试用例集文件；点击“Choose Expected Output”，输入测试用例集对应的预期输出文件。点击各自按钮转到文件选择框，选择所需要的文件。如图 3-3 所示：

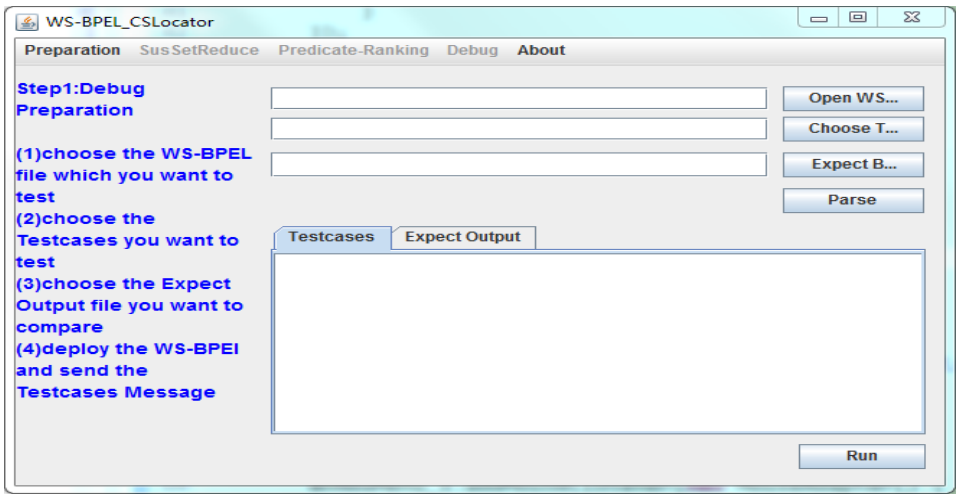

图 3-2 WS-BPEL 文件选择界面图

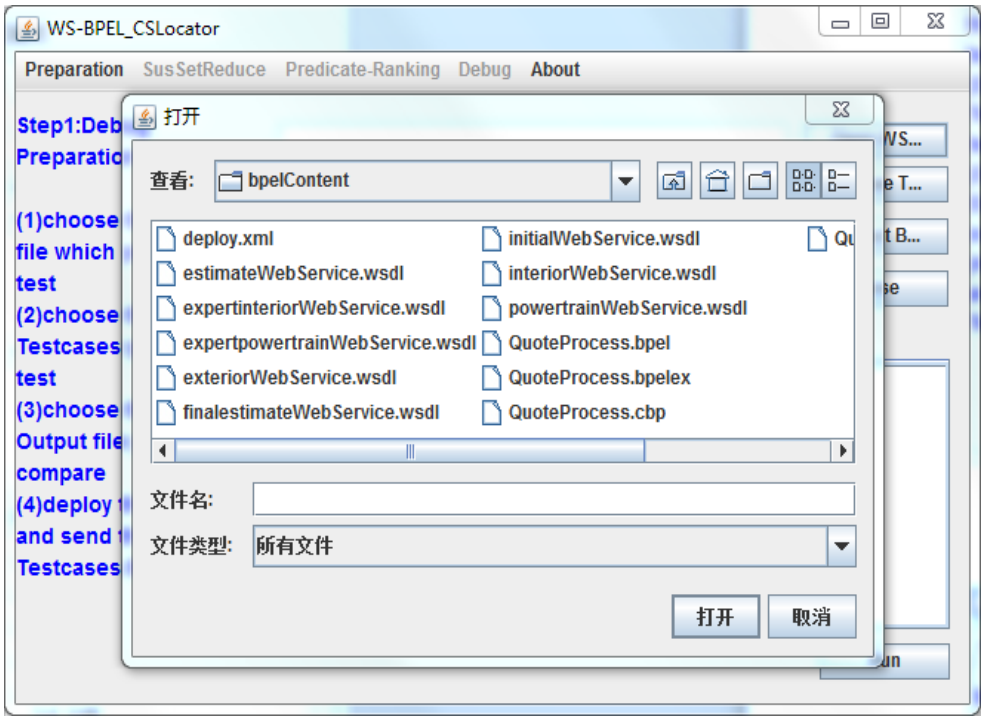

图 3- 3 打开文件界面图

选择了 WS-BPEL 文件之后，点击“Parse”，可以得到待测的 WS-BPEL 程序的结构，如图 3-4 所示。选择了测试用例集和预期输出之后，可以从 “Testcases”和“Expect Output”选项框中看到选中的文件内容，如图 3-5 和图 3-6 所示。点击“Run”之后，启动 tomcat 服务器，对 WS-BPEL 服务进行部署，执行测试用例集并记录执行轨迹，得到的执行轨迹如下图 3-7 所示。

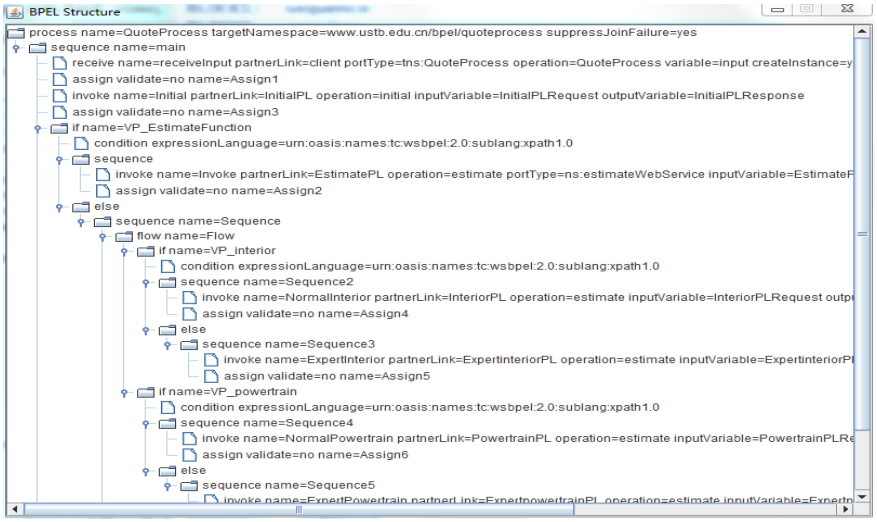

图 3- 4 程序解析图

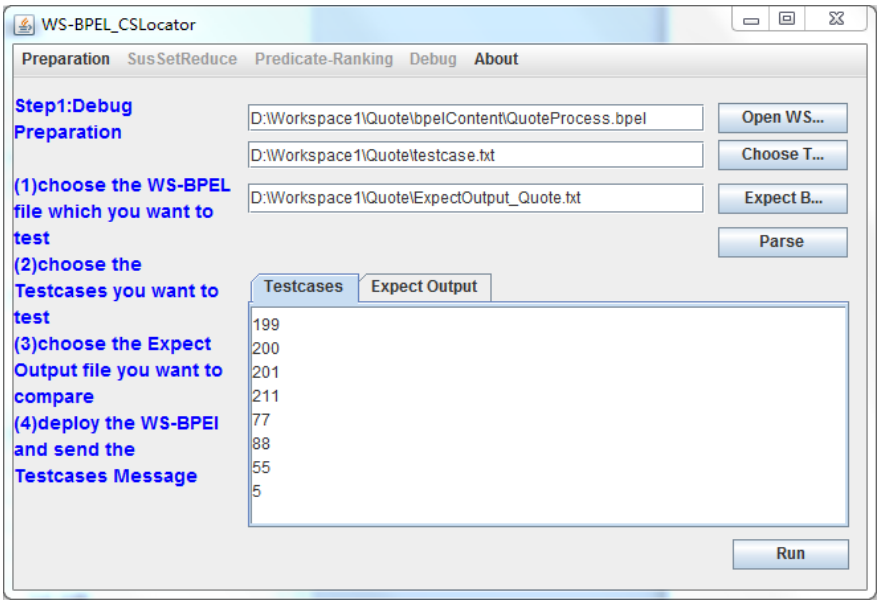

图 3- 5 测试用例显示图

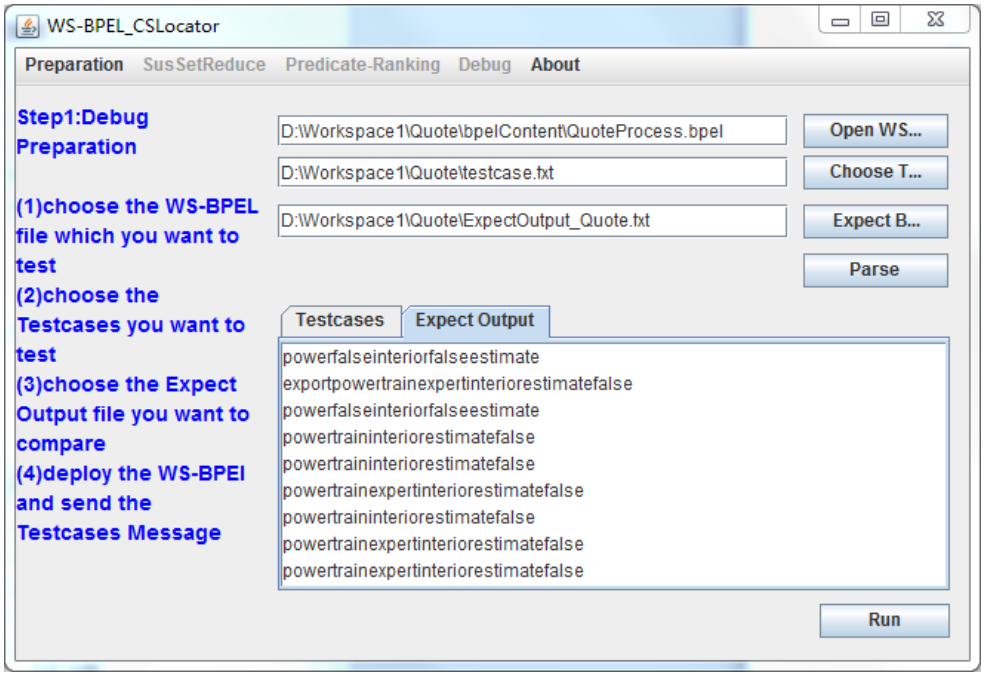

图 3- 6 预期结果显示图

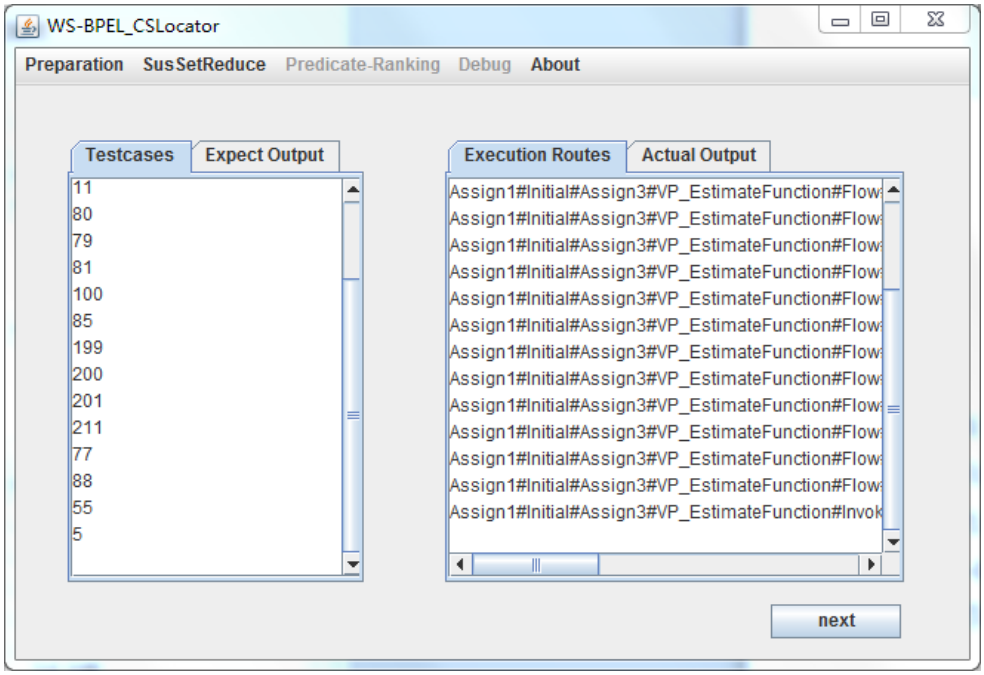

图 3- 7 执行结果显示图

第二步：根据得到的执行轨迹和预期结果比较，将执行轨迹集分为失败执行轨迹 FRS 和成功执行轨迹 PRS，如下图 3-8 所示。“Nodes”文本框中显示的是可疑集合中的结点。点击“PreBranch”按钮，如果该谓词在失败执行轨迹 FRS 中真分支和假分支数量都大于 1，删除可疑集合中控制依赖于该谓词的结点，如下图 3-9 所示。点击“Atom Activity”按钮，如果 PRS 中执行轨迹的数量大于 0，删除可疑集合中公共原子活动，如下图 3-10 所示。

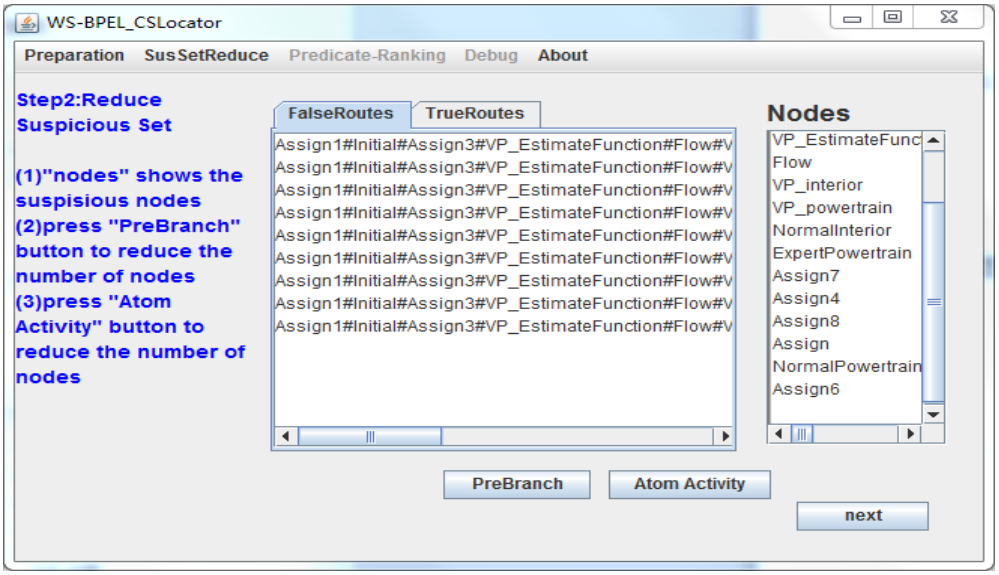

图 3- 8 可疑集合显示图

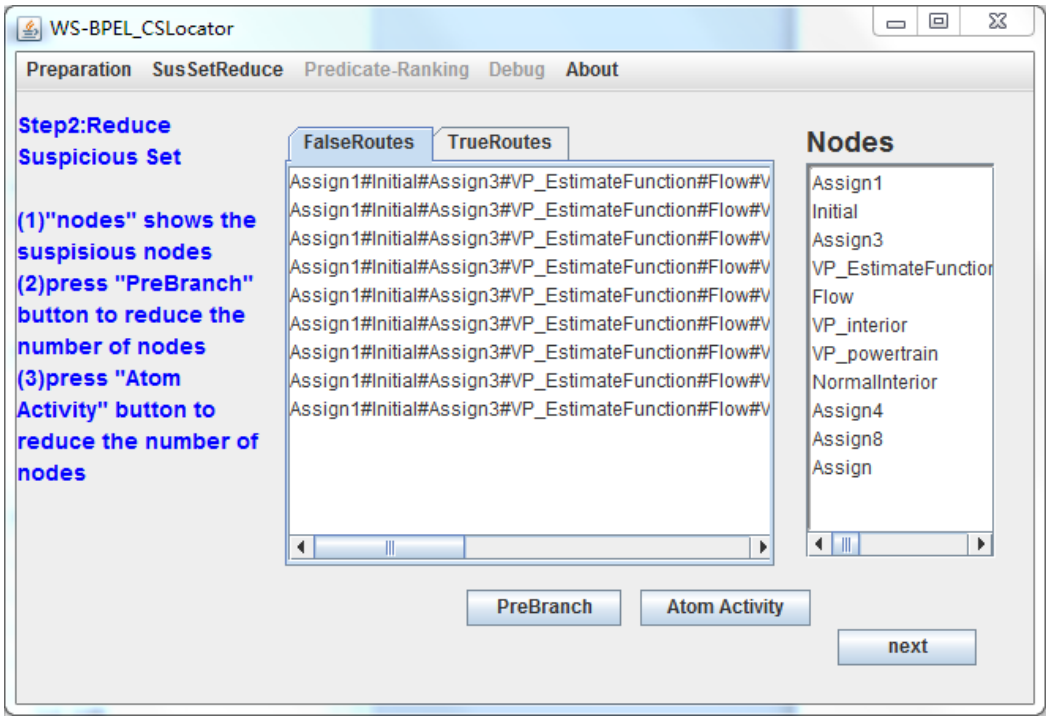

图 3- 9 判断谓词分支结果显示图

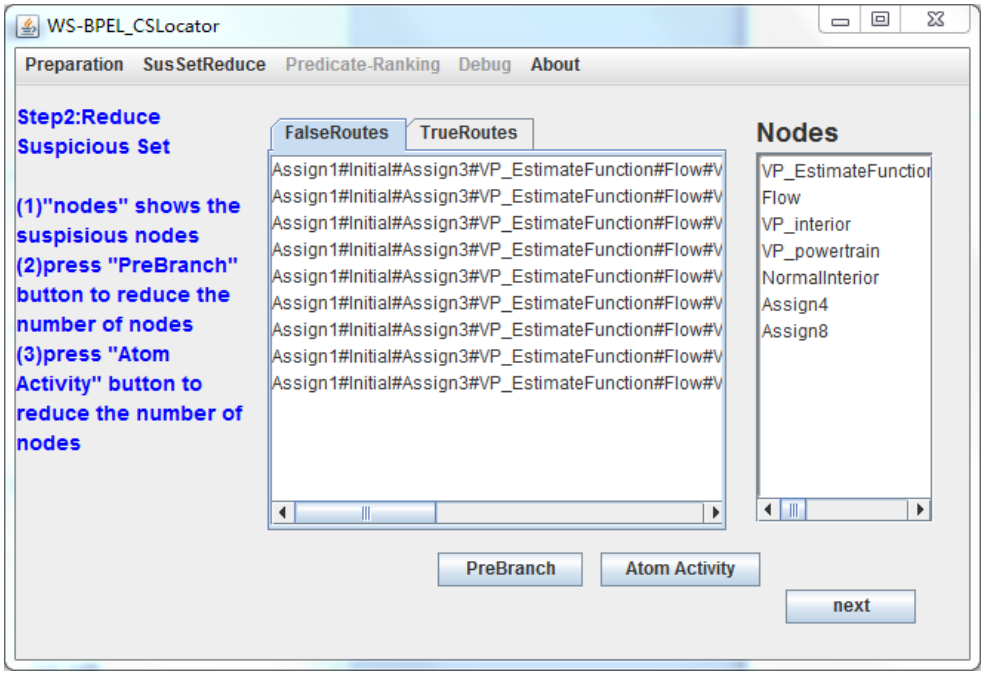

图 3- 10 判断公共原子活动结果显示图

第四部：根据 SOBER 故障定位技术中的公式对可疑集合中的谓词进行排序。统计该谓词在失败执行轨迹 FRS 和成功执行轨迹 PRS 中表达式取真值的次数，根据 SOBER 算法中的公式计算怀疑度，并根据怀疑度由高到低对谓词进行排序，

如下图 3-11 所示。

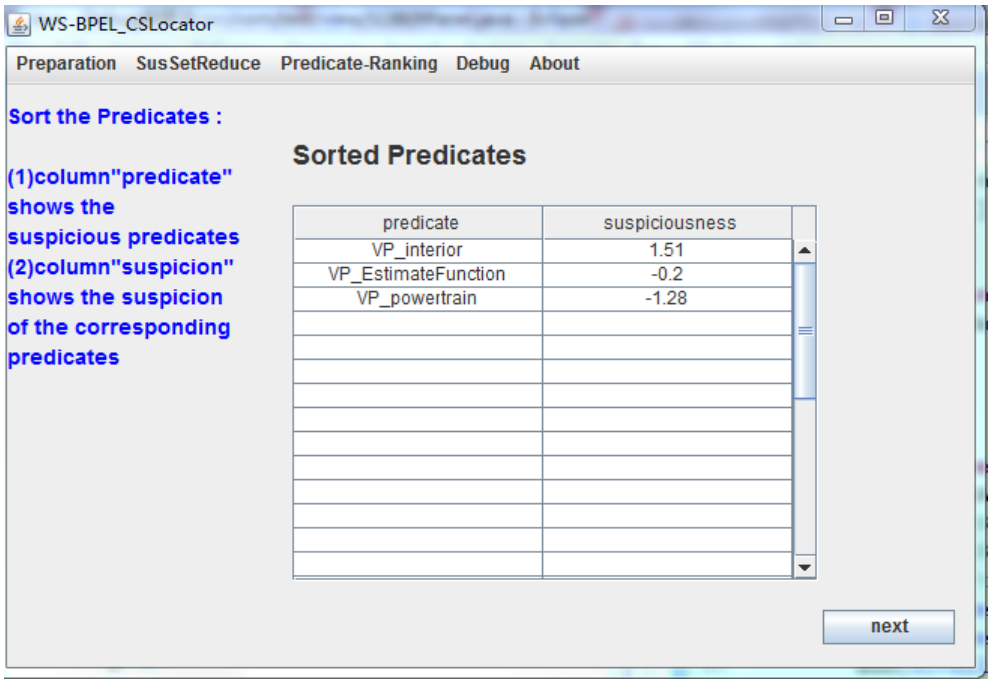

图 3- 11 谓词怀疑度排序结果图

第五步：根据上述的可疑集合缩减规则和谓词排序的结果，对可疑集合中的结点进行排序。根据谓词的怀疑度从高到低输出可疑集合中的谓词结点，然后输出可疑集合中的非谓词结点，得到定位结果报告，如下图 3-12 所示。

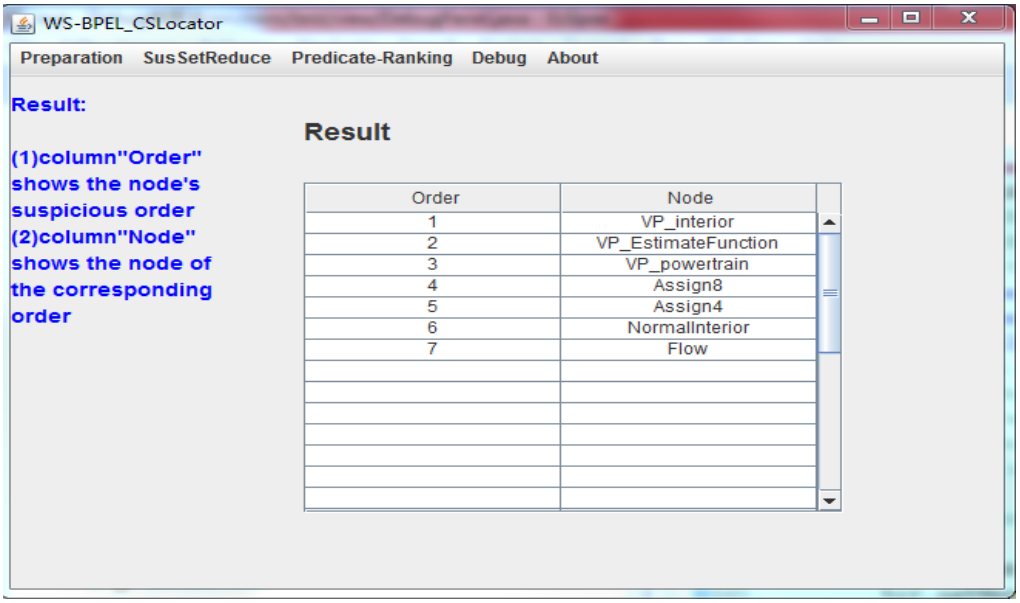

图 3- 3 定位结果报告图