模型驱动的SOA软件测试与监控技术研究

项目进展报告——20161228

1. 整体思路：

图1所示为整体思路框架图



* 首先根据服务组合程序的设计需求及该组合程序基于的WSDL文档生成相对应的行为模型（DBM）；
* 解析BPEL源程序，生成服务组合程序实现对应的行为模型（IBM）；
* 静态检测实现对应的行为模型（IBM），报告BPEL程序实现过程中违反的约束，生成静态检测报告；
* 遍历行为模型（DBM）生成测试路径，获取该路径中的约束，使用约束求解器生成测试用例；
* 在扩展后的ODE引擎上使用生成的测试用例执行相应的BPEL程序，获取违反约束的测试用例执行，生成测试报告。

1. WSDL文档约束提取扩展

理论上，服务应该是无状态的，然而，实践中，一个服务内部的操作在开发时往往存在一定的控制流约束关系。例如，电子支付系统中的“转账”操作依赖于“建立账户”这样的操作。此外，还有些操作依赖于特定的数据状态，即根据上下文数据决定是否执行相应的操作。

基于上述原因，我们从**控制流**与**数据流**两个角度对同一个服务提供的**操作间的约束**进行提取：

* **控制流**

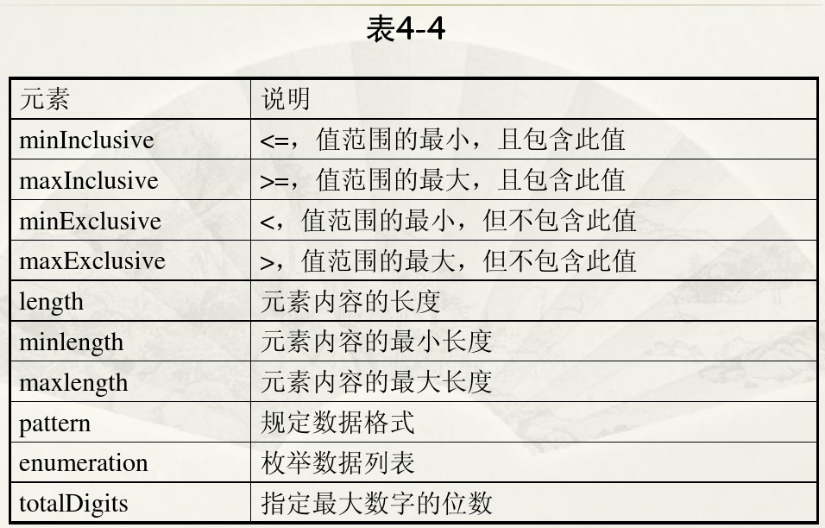
**序列约束：一个操作的执行前置条件必须为另一个操作执行**

**消息约束：某个响应消息响应的前置条件**

* **数据流**

**参数约束：**

**操作自己的输入参数范围约束，该约束中使用的字段依据schema结构元素声明**



**计算约束：**

**操作A的输出或执行过程影响的变量，参与到了操作B的计算中，虽然直接执行操作B没有控制流上的错误，但是返回结果与先执行A再执行B后的结果不同。**

1. 服务行为约束的表示（约束示例）：

本章节以航空行李计费系统，和ATM系统为例介绍扩展的WSDL文档中的约束表达。

**序列约束：**

我们以ATM服务提供的登录操作与转账操作为例，转账操作必须在登录操作执行后（且登录成功）才可以执行，该约束表达如下所示：

|  |
| --- |
| <wsdl:operation name="transfer">  **<wsdl:restriction preOp="login" preCT="loginResponse=true" />**  <wsdl:input message="tns:transferRequest"></wsdl:input>  <wsdl:output message="tns:transferResponse"></wsdl:output>  <wsdl:fault name="fault01" message="tns:InvalidAccountID"></wsdl:fault>  <wsdl:fault name="fault03" message="tns:InvalidAmount"></wsdl:fault>  </wsdl:operation> |

使用restriction标签对操作进行限制描述，preOp标签标识该操作的前置操作，preCT标签标识前置操作执行之后的状态，当用户已经登录账户并且成功后，可进行转账操作。

**消息约束：**

我们以航空行李计费系统为例，该服务提供一个feeCalculation操作，该操作输入消息为feeCalculationRequest输出消息为feeCalculationResponse，错误消息为MACException，当出入的参数chooseAirClass在[0,3]以外，返回错误消息。

|  |
| --- |
| <wsdl:operation name="feeCalculation">  <wsdl:input message="impl:feeCalculationRequest" name="feeCalculationRequest">  </wsdl:input>  <wsdl:output message="impl:feeCalculationResponse" name="feeCalculationResponse">  </wsdl:output>  <wsdl:fault message="impl:**MACException**" name="MACException">  </wsdl:fault>  </wsdl:operation>  <wsdl:message name="MACException">  <**wsdl:restriction reCT="chooseAirClass<0 || chooseAirClass>3"/**>  <wsdl:part element="impl:fault" name="fault">  </wsdl:part>  </wsdl:message> |

使用restriction标签对消息进行限制描述，reCT标签标识该消息执行的约束状态，当出入的参数chooseAirClass在[0,3]以外，返回错误消息。

**参数约束：**

我们以ATM服务提供的转账操作为例，转账操作中转账金额amount操作，其限制如下所示：

|  |
| --- |
| <xsd:element name="**transferRequest**">  <xsd:complexType>  <xsd:sequence>  <xsd:element maxOccurs="1" minOccurs="1" name="accountFrom" type="tns:accountNumber"></xsd:element>  <xsd:element maxOccurs="1" minOccurs="1" name="accountTo" type="tns:accountNumber"/>  <xsd:element name="mode" type="tns:modeType"></xsd:element>  <xsd:element name="**amount**" type="**tns:amountType**"></xsd:element>  </xsd:sequence>  </xsd:complexType>  </xsd:element>  <xsd:simpleType name="**amountType**">  <xsd:restriction base="xsd:int">  <xsd:minInclusive value="100"/>  <xsd:maxInclusive value="5000"/>  </xsd:restriction>  </xsd:simpleType> |

使用simpleType标签标记对参数的限制，restriction标签描述了参数的约束，其中amount操作为int型操作，其输入范围为[100,5000]。

**计算约束：**

|  |
| --- |
| <wsdl:operation name="B">  **<wsdl:restriction caOp="A"/>**  </wsdl:operation> |

使用restriction标签对操作进行限制描述，caOp标签标识该操作的计算过程中使用到的变量受到哪些操作的影响。

1. 错误模糊定位

执行基于约束生成的测试用例，如果一个失败的测试用例发生，适当的将尝试定位的一部分。

依据建立的行为模型，可对服务组合程序的实现进行静态检测。创建客户端后，可对服务组合程序进行动态执行，获取执行过程信息及结果，对其分析进行动态检测。

我们将发生的不同错误划分为两类：**组合程序错误**与**原子服务错误**

* **组合程序错误**

1. 服务组合程序的操作调用顺序不满足序列约束（静态检测）
2. 调用参数不匹配：
3. 参数赋值错误（组合程序需求为a1=b1,a2=b2结果实现为a1=b2,a2=b1）（静态检测）
4. 使用人员输入的参数不符合该操作的约束要求（动态检测）
5. 一个操作的输出不匹配另一个操作的输入约束（动态检测）
6. 调用序列错误：某个测试用例应该执路径1，但是现在执行了路径2（多为判断分支导致）（动态检测）

* **服务错误**

1. 服务执行前未出错，但其执行结果与约束不匹配（动态检测）
2. 调用的服务不可用（动态检测）
3. 测试数据生成

有效的数据验证（约束求解）（这里提及的有效是指的可以执行服务组合程序）

无效的数据（依据服务组合程序的接口，使用黑盒测试技术直接生成）