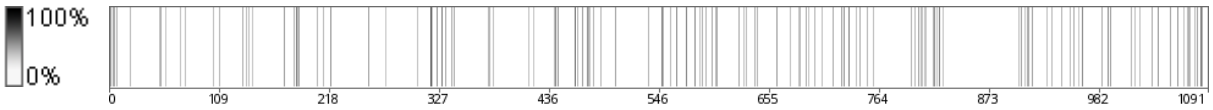
PaperPass旗舰版检测报告
简明打印版

比对结果(相似度):

总　体： 8% (总体相似度是指本地库、互联网的综合对比结果)
本地库： 8% (本地库相似度是指论文与学术期刊、学位论文、会议论文、图书数据库的对比结果)
期刊库： 5% (期刊库相似度是指论文与学术期刊库的比对结果)
学位库： 5% (学位库相似度是指论文与学位论文库的比对结果)
会议库： 1% (会议库相似度是指论文与会议论文库的比对结果)
图书库： 3% (图书库相似度是指论文与图书库的比对结果)
互联网： 0% (互联网相似度是指论文与互联网资源的比对结果)

编　号：5A1035495523B5ZV4
版　本：旗舰版
标　题：删除图片查重
作　者：贾婧婷
长　度：34982字符(不计空格)
句子数：1091 句
时　间：2017-11-18 21:27:37
比对库：学术期刊、学位论文、会议论文、书籍数据、互联网资源
查真伪：http://www.paperpass.com/check

句子相似度分布图:



本地库相似资源列表(学术期刊、学位论文、会议论文、书籍数据):
暂无本地库相似资源

互联网相似资源列表：
暂无互联网相似资源

全文简明报告:

{55%：行为模型驱动的服务组合程序测试用例生成技术}

本节主要通过示例介绍行为模型驱动的服务组合程序测试用例生成技术的方法简述、关键技术及具体实现。

{52%：行为模型驱动的服务组合程序测试方法框架}

本文采用服务行为约束刻画由于服务存在隐含行为逻辑导致服务调用失效的情况,

{49%：扩展 Web服务描述语言（ WSDL），显式描述服务正确使用的数据约束和控制约束，} 解决服务规格说明中缺乏服务行为逻辑描述而导致服务错误调用的问题。 {48%：提出了行为模型驱动的服务组合程序测试用例生成技术。} 图3-1所示为行为模型驱动的服务组合程序测试用例生成技术框架图，主要包含五个部分： {46%：解析扩展WSDL文档、服务行为模型图建立、测试序列生成、测试用例生成及测试执行。}

图3-1 技术框架图

扩展WSDL解析（EX-WSDL Paring）： 解析服务开发人员提供的扩展 WSDL、获取Web服务提供的操作及其约束（ Constraint Result，用于行为模型生成）、调用操作的SOAP消息框架（ SoapEnv， 用于测试数据填充）及 XML结构定义文档（ XSD，用于消息校验）；

行为模型构建（Behavior Model Construction）： 针对提取出的约束文档（Constraint Result）生成该服务的行为模型（Model）；

测试序列生成（Test Path Generation）： 针对生成的行为模型使用定义的覆盖准则，生成测试序列（Test Seqs）；

测试用例生成（Test Case Generation）： 根据测试序列，从决策表中获取符合该序列的执行约束，使用 z3求解器求解出测试数据， 填充到 SOAP消息中形成可执行的测试用例（ Test Suite）；

测试执行（Test Execution）： {43%：集成SoapUI模拟客户端，执行测试用例，生成测试报告（Test Report）；}

本课题重点研究以下四个问题：

WSDL文档行为约束扩展： 如何定义与描述服务的行为约束；

基于扩展WSDL文档的Web服务行为模型生成方法： 定义了服务约束并扩展WSDL文档后，解决如何解析给定的扩展WSDL、如何构建与存储Web服务行为模型；

行为模型驱动的测试用例生成： {49%：Web服务行为模型构建完毕后，考虑基于建立模型的测试序列、测试数据与测试用例的自动生成。} 主要解决如何从给定行为模型中生成测试序列、如何根据测试序列求解相对应的测试数据以及如何将测试数据与测试序列结合，生成可执行的测试用例；

测试执行与结果判定： 测试与监控服务的运行以及对服务操作的调用是否符合约束条件进行判断，针对执行错误的测试用例，尝试定位违反的束类型。

接下来将分节介绍解决上述问题的关键技术及其具体实现。

WSDL文档行为约束扩展

{51%：由于服务使用者只能依据规格说明访问服务。} 然而服务规格说明仅仅包含了服务接口说明（操作包含哪些数据以及数据格式要求），但是仅描述这些结构化的语法信息是不够的， 参与组合的服务通过接口中的多个操作提供服务，这些操作潜藏的各种数据流与控制流方面的约束往往是业务处理的关键。 控制流约束表示操作之间的执行顺序约束，例如电子支付系统中转账操作依赖于建立账户这样的操作之后。 数据流约束表示触发某个特定操作所需的特定数据状态，根据上下文数据决定是否执行相应的操作， 例如， ATM转账

业务中，当转账的金额不大于卡内余额时可触发转账操作。 同时对于同一操作内部来说，输入数据范围的不同可能导致该操作的内部执行逻辑的不同。 然而这类约束通常采用自然语言进行描述，且与 Web服务规格说明（ WSDL）分离， 因此需要考虑如何形式化的刻画与表达服务行为约束、如何建立 WSDL文件与服务实现之间的关系， 解决 Web服务实现与规格说明分离的问题。

本节将介绍支持的约束类型的定义和应用场景及如何扩展WSDL支持约束的表达

行为约束类型定义与描述

通过阅读文献研究总结了如下六个Web服务提供操作自身及相互之间可能存在的约束：时效约束[30]、序列约束[24，31，39，40，41，57]、调用约束[23]、参数范围约束[31， 57，58，59]、参数关系约束[4，58]、区域约束[30]。 如图3-2所示，按照约束所在层次分为：{42%：服务层次约束、操作层次约束，其中操作层约束又分为数据流与控制流约束。}

图3-2 服务约束定层次与分类

{49%：本文使用扩展巴科斯范式(Extended Backus- Naur Form)[60]表述行为约束 Constraint， EBNF是巴科斯的基本范式（ BNF）元语法符号扩展，} 在 BNF基础上主要扩展了如下规则；

终结符被严格的包围在引号 ”…” 或 ’…’ 中，给非终结符的尖括号”[…]”可以省略；

进一步提供了重复次数（*、+、[]），排除选择和注释等增强机制。

EBNF可以用来表示一个上下文无关文法，其基本符号及意义如表3-1所示。

表3-1 EBNF基本符号及意义

行为约束Constraint语法如下所示：

Constraint标识行为约束，使用 JSON[61]键值对形式表示，包含参数关系约束（paraRelation）、区域约束（ ipRegion）、调用约束（ invokeOp）、序列约束（preOp、 Iteration）、时效约束（ eTime）；

ValuePR标识参数关系约束内容，为多个 Relationship组成的 JSONArray，Relationship标识两个参数之间的关系， 由参数名称及关系符号组成，其中RelationSymbol支持=、]、[、]=、[=、！ =关系的表达，参数名称由参数所在操作名（OpName）.参数名（OpParameter）组成，OpName及OpParameter使用JAVA变量定义规则：以字母、数字、下划线及美元符组成；

ValueIR标识操作区域约束，为可调用操作的 IP地址（ IpAdress）范围，IpAdress标识一个以点分十进制表示法表示的 IPv4地址， IpField标识其中一个字节，可选数字范围为0-255；

ValueIO标识某操作调用的其他操作，为多个操作名（OpName）组成的JSONArray；

ValuePO标识操作正确调用顺序依赖，使用正则表达式模式表示其顺序依赖，支持重复（*，+）与替代关系（ ）表达。

ValueI标识操作是否可重复调用，使用true或false进行标记；

eDate标识服务时效约束，为yyyy-MM-dd格式的日期值；

下面使用一个停车计费系统实例，进行Web服务行为约束的举例。

例3-1： 假设一个停车计费服务，具有入库（login）及出库计费（feeCalculate）操作，其中出库计费操作需要在入库操作执行成功后才能执行。

对于login操作，输入参数为车牌号(License)及入库时间（loginTime），其约束如表3-2所示。

表3-2 login操作参数及其范围约束

{42%：feeCalculate操作根据司机的车辆类型、停车日期、停车时间计算停车费用。}输入参数及其约束如表3-3所示。

表3-3 出库计费操作参数及其范围约束

其中License代表出库计费车辆车牌号，应该与入库时相同； type代表车辆类型，由枚举值{0，1，2}分别代表不同类型车辆； timeout代表车辆出库时间，停车时间为timeout-loginTime，因此要求timeout大于等于loginTime； dayOfWeek代表停车日是否为工作日；discountCoupon表示是否使用优惠券。

具体示例如下所示：

时效约束（eTime Constraint）

需求的快速变化导致服务面临经常性的修改，被调用服务接口可能处于停用或维护状态，考虑添加时效约束， 约束服务有效时间，超过有效期可能存在由于服务更新导致的 WSDL与服务实现不匹配等问题。

假设服务A的预计有效时间为2018.01.31，即服务提供商预计在2018年1月31日前不会对服务A进行修改或停用，其时效约束为：

”eTime”： ”2018-01-31”

序列约束

序列约束约束了操作执行的顺序依赖及操作是否可以重复执行。 其中某操作的顺序约束（preOp Constraint）指定操作正确执行前需要执行的操作顺序； 重复调用约束（Iteration Constraint）指明操作执行成功后是否可以重复继续执行该操作。

针对例3-1出库计费操作，其顺序依赖约束为：

”preOp”：
”((login)(loginResponse_Success)(feeCalculate)(feeCalculateResponse_Success))*(login)(loginResponse_Success)”

针对入库操作，若其执行成功，表明该车辆已入库，则无法对该车辆再一次进行入库操作，因此针对入库操作，其重复调用约束为：

”Iteration”: ”false”

调用约束（invokeOp Constraint）

一个服务操作可调用其他操作用以完成自己的任务，但该操作执行过程可能由于违反被调用操作约束而发生错误，因此表示与追踪这种调用关系十分重要。 调用约束表示了操作之间的调用关系。

{41%：假设，操作A的执行过程需要调用操作B与C，那么操作A的调用约束为：}

”invokeOp”: [”B”，”C”]

参数关系约束（paraRelation Constraint）

参数关系约束约束不同服务操作的输入参数的关系，有些操作之间的输入参数存在约束关系，调用操作时，即使输入参数符合服务操作的 XMLSchema约束， 但是存在由于违反了操作间的参数关系约束而导致操作调用失败的可能。 {44%：由此引入参数关系约束，描述正确使用服务操作时参数与其他相关服务操作的参数的关系。}

针对例3-1出库计费操作，可知其出库车牌号与入库车牌号应该相同，出库时间大于等于入库时间，因此，该操作的参数关系约束为

”paraRelation”: [”feeCalculate.License = login.License”，”feeCalculate.timeout]= login.loginTime”]

参数范围约束（paraRestriction Constraint）

Web服务可能要求在实现某项操作时用户输入的某些数据必须满足特定的取值范围。 由此引入参数范围约束。 描述输入数据的数据类型和取值约束当服务使用者输入违反该约束时，调用服务出错。

由于WSDL使用XSD定义消息交换所需的各种数据类型，因此本文使用XSD限定（restriction）标签约束参数范围。 {57%：限定（restriction）用于为XML元素或者属性定义可接受的值，本文支持的数据类型的限定及其描述如表3-4所示。}

表3-4 参数范围约束限定

针对例3-1，图3-3展示了 feeCalculate操作参数范围约束实例，其中图 a）定义了一个带有枚举限定的名为 type的整型参数， 可接受的值只有0、1、2； 图b）定义了一个带有模式限定的名为License的字符串参数，可接受以BJA-BJY开头，5为数字结尾的字符串； 图c）义了一个带有上下界限定的名为timeout的整型参数，可接受的值为[0，24]，包括0和24； 图d）定义了一个带有上下界限定的名为id的整型参数，可接受的值为(0，10)，不包括0和10。

图3-3 参数范围约束实例

区域约束（ipRegion Constraint）

某些服务提供的操作仅仅能够在特定网络访问，区域约束定义了访问服务操作的IP地址范围。

假设某个操作A只有特定网段202.204.62.0到202.204.62.255拥有访问权限，则该操作的区域约束为：

”ipRegion”： ”202.204.62.0-202.204.62.255”

为准确描述约束类型的定义，我们使用语法图[62]。 {47%：图形化描述行为约束文法，行为约束语法图如图3-4所示。}

图3-4 行为约束语法图

扩展WSDL支持行为约束的表达

{41%：Web服务约束定义完成后，考虑如何在服务描述中存储包含约束的服务行为，即如何支持服务行为约束的表达。}

现有两种方法向Web服务描述中添加行为信息，一是扩大现有WSDL协议，另一种是实现一个能够表达行为约束的新的协议。 考虑到当前的 WSDL作为服务描述的标准规范广泛应用于 Web服务描述且新协议没有统一的标准， {42%：本文采取第一种方法，通过扩展WSDL支持服务行为约束的表达。}

现有研究大部分依赖 XML可扩展标记，在 WSDL中定义新的标签及命名空间，由于新定义标签众多且没有统一标准， 本文考虑使用 WSDL标准中包含的[documentation]元素支持服务行为约束表达。 如图3-5 a)所示， WSDL[2]使用可选的[documentation]元素，作为服务开发和使用人员可读文件的容器， 元素的内容是任意的文本，且[documentation]元素被允许嵌套在任何其他 WSDL标签元素中。[documentation]标签作为WSDL标准中的元素，出于其灵活性与广泛使用，本文使用[documentation]标签作为服务行为约束表达方式。

图3-5 WSDL扩展前后元素对比

本文针对标准 WSDL服务层与操作层进行扩展生成扩展后 WSDL（记作 EX-WSDL）， 将[service]与[operation]标签内的可选元素[documentation]定为必选元素， 向其中加入本文定义的行为约束，如图3-5 b)所示，在其中添加服务及操作的相关行为约束。 例3-1（停车计费服务）扩展后的WSDL文档如图3-6 b)所示，该服务操作输入Schema格式如图3-6 a)所示：

a) 停车计费服务入库及出库计费操作参数范围约束

b) 停车计费服务行为约束扩展

图3-6 加入行为约束后的停车计费服务EX-WSDL实例

基于EX-WSDL文档的Web服务行为模型生成方法

{43%：定义了服务约束并扩展WSDL文档后，考虑基于服务行为的形式化描述模型的建立。}本课题采用事件序列图(ESG)[63]对服务行为进行建模，重点关注服务进行了何种操作，操作的数据状态及操作之间的依赖。 事件序列图提供一种简洁、易于理解的形式化表达手段，由节点及边组成，节点代表触发的事件， 边代表事件的转移，同时具有较强的理论技术，能够表达数据约束及控制依赖，有如下定义[29]。

事件序列图(ESG)定义为2元组ESG =[V， E]。 V表示ESG中全部（有限）节点集合，代

表用户输入或系统响应的事件且V； V，代表ESG中有限入口节点集合； V，代表ESG中有限出口节点集合； E表示ESG中的全部（有限）边集合，代表事件的有向转移EVV；

事件序列(ES)： 对于一个序列[v0，，vk]，如果(vi ，vi+1)E， i=0，，k-1，则称该序列是事件序列；

完整事件序列(CES)： 对于一个事件序列，若其起点为入口节点且结束节点为出口节点，则称该事件序列为完整事件序列。

我们对事件序列图进行扩展，定义 Web服务行为模型（ WSBM）， 下面我们介绍WSBM相关的基本定义并通过例3-1进行举例说明（例3-1为停车计费服务（ ParkingFeeCalculator）其行为模型如图3-8所示， 对应的 WSDL如图3-6所示）：

定义1（Web服务行为模型WSBM）： WSBM是一个有向图模型，表示为四元组WSBM=[N，D，V，E]。 其中，N标识该模型名称，表示该模型所描述的服务； D标识该模型有效期限，从服务时效约束中提取； {52%：V为WSBM中非空节点集合，E为WSBM中有向边集合，表示点的转移。} DT标识服务提供操作的决策表集合。

定义2（模型节点）： V为WSBM中非空节点集合，V={v0，，vn}，n为全部节点数减1。对于任意节点 v0，， vn(0 i n)表示为六元组 vi=[N， I， C， B，A， T，]，其中 N表示 vi的名称； I表示vi的编号； C表示vi的约束； B表示vi的前继节点集合； A表示vi的后续节点集合； {46%：T表示vi的类型，模型图中定义了不同类型的节点及约束，节点类型包括：} 开始节点（ Start）、初始节点（ Init）、结束节点（ End）、请求节点（ Req）、响应节点（ Res）， {62%：每种类型的节点及其具体含义如表3-5所示。} 其中，WSBM只有一个入口节点、一个出口节点，即节点集合中仅有一个开始节点与一个结束节点。 请求节点的约束类型包括： {51%：参数范围约束、参数关系约束、区域约束、顺序约束、调用约束；}

定义3（模型边）： E为WSBM中有向边集合，代表事件的有向转移EVV，E={e0，，em}，m为全部边数减1。 对于任意边ej (0 j m)表示为三元组ej=[N，TO，FR]。 其中N表示ej的名称； TO表示ej的起始点； FR表示ej的终端点；

定义4（前继节点preNode）： vi是vj的前继节点记作vi=preNode(vj)，当且仅当满足以下条件：

viV

vjV

(vi ，vj)E

定义5（后继节点nextNode）： vi是vj的后继节点记作vi=nextNode(vj)，当且仅当满足以下条件：

viV

vjV

(vj ，vi)E

定义6（节点相关性Correlation）： {44%：vi与vj的互为相关节点，记

作Correlation(vj ，vi)，当且仅当满足以下条件：}

viV

vjV

(vi ，vj)E或(vj ，vi)E

表3-5 Web服务行为模型节点类型定义

图3-7 服务行为模型的UML类图

{42%：服务行为模型的 UML类图如图3-7所示，其中 Graph类为 Web服务行为模型实现类，} 由 Node类与 Edge类组成， Graph类包含名称（ Name）、时效（ Date）、节点集合（ nodes）及边集合（ edges）属性。 Node类由 Constraints类及 Condition类组成，包含节点名称（ name）、编号（ id）、类型（ type） 、约束（ constraints、 condition）、前继节点集合（ beforeNodes）及后继节点集合（ afterNodes）属性。 Constraints类定义某个节点的参数关系约束（paraRelation）、调用约束（invokeOp）、顺序约束（preOp）及重复调用约束（Iteration）。 Condition类存储节点执行所需参数名称（name）、类型（type）及限制（maxExclusive、minExclusive、maxInclusive、minInclusive、pattern、enumeration）。

图3-8显示了图3-6停车计费服务的EX-WSDL的行为模型，模型节点为白色顶点，内部为节点标号； {55%：边用黑色箭头表示，箭头初始为边的起始点，指向为边的终端点。} 图3-9显示了行为模型中请求节点v2与v5的行为约束。

图3-8 例3-1停车计费服务行为模型图

图3-9 节点行为约束

根据定义有：

WSBM=[N，D，V， E]，其中： N=ParkingFeeCalculator； D=2018-01-31；V={v0，v1，v2，v3，v4，v5，v6，v7，v8}；

E={e0，e1，e2，e3，e4，e5，e6，e7，e8，e9，e10，e11}；

节点v0=[N，I，C，B，A，T，]，其中N=Start； I=0； C=null； B=null；A={v1}； T=Start； v0=preNode(v1)； v1=nextNode(v0)；

节点v2=[N，I，C，B，A，T，]，其中N=login； I=2； B={v1，v6}； A={v3，v4}；T=Req； 该节点行为约束如图3-9所示，由图3-6 a)参数范围约束与图3-6 b) login操作documentation标签内容解析；

节点v3=[N，I，C，B，A，T，]，其中N=loginResponse_succ； I=3； C=null；B={v2}； A={v5，v8}； T=Res；

解析扩展WSDL生成服务行为模型的过程可总结为图3-10所示算法

图3-10 Web服务行为模型生成算法

以图3-6所示EX-WSDL转换为图3-8所示行为模型的过程为例，描述该算法执行过程。 该算法由四个主要步骤组成：

初始化（1-3）： 初始化一个行为模型 g（ Nodes=、 Edges=），将其名称属性设置为 EX- WSDL URL中解析出的服务名称， 将其有效时间设置为服务时效约束中的 eTime。 向Nodes中添加一个start节点、init节点与end节点，并设置节点属性。{51%：将start节点添加进init节点的前继节点，init节点添加进start节点的后续节点。}针对图3-6有： g.name= ParkingFeeCalculator； g.endTime=2018-01-31； Edges=，Nodes={v0，v1，v8}； v0.name=start； v0.id=0； v0.type=Start；v1.name=init； v1.id=1； v1.type=Init； v0=preNode(v1)； v1=nextNode(v0)。

模型节点元素添加（4-12）： 解析EX-WSDL，获取其全部操作集合OpSet。 针对该集合中的每个操作op，向Nodes中添加该操作相关的请求节点与响应节点并设置节点属性； 解析操作Schema各式设置请求节点参数范围约束，解析操作documentation标签内容设置请求节点其他行为约束； 建立请求节点与响应节点前后继关系、响应节点与结束节点前后继关系。针对图3-6有： 停车计费服务提供login与feeCalculate操作。 针对login操作有： 添加请求节点v2，其中v2.name=login； v2.type=Req； v2.id=2； 解析login操作的documentation标签内容获取v2的行为约束（如图3-9所示）；v3.name=loginResponse_succ； 添加操作调用成功时的响应节点v3，其中v3.type=Res；v3.id=3； 由于login操作的重复调用约束为false，因此当操作响应成功后并不能直接调用该操作，因此无需设置v3为v2的前继。 添加操作调用失败时的响应节点v4，其中v4.type=Res； v4.id=4； 最后设置响应节点与结束节点的相关性，即v3=preNode(v8)；v4=preNode(v8)； v8=nextNode(v3)； v8=nextNode(v4)。 针对操作feeCalculate的处理过程相同，不做赘述。 此时Nodes={v0，v1，v2，v3，v4，v5，v6，v7，v8}。

序列关系建立（13-22）： 针对行为模型g中所有类型为请求类型的节点，解析该节点顺序约束，建立与该顺序约束相关节点之间的前后继关系。 若该节点无顺序约束，则建立该节点与init节点的前后继关系。 针对Nodes有请求类型节点集合为{v2，v5}。 由图3-6可知，v2无顺序约束，因此建立v2与初始节点init关系，有init=preNode(v2)，v2=nextNode(init)。 解析v5节点的顺序约束可知：v3=preNode(v5)； v5=nextNode(v3)； v6=preNode(v2)； v2=nextNode(v6)，由此建立相关节点间的前后继关系。

模型边元素添加（23-27）： {41%：针对行为模型g节点集合Nodes中所有节点，遍历该节点所有后继节点，向Edges集合添加一条从该节点到其后继节点的边并设置其属性。} 如针对节点v0，其后继节点集合为{v1}，因此向Edges集合添加e0，e0.name=e0； e0.from=0；e0.to=1； 针对节点v2，其后继节点集合为{v3，v4}，因此向Edges集合添加{e2，e3}，其中e2.name=e2； e2.from=2； e2.to=3； e3.name=e3； e3.from=2； e3.to=4；针对其余节点的处理过程相同。 此时有， Edges={ e0， e1， e2， e3，e4， e5， e6， e7， e8， e9， e10， e11}。

根据以上四个步骤，可生成停车计费服务的Web服务行为模型，该模型如图3-8。

{80%：行为模型驱动的测试用例生成技术}

{41%：理论上，服务应该是无状态的（即服务提供的操作可以以任意顺序调用）。} 然而实践中，这些操作间存在一定的数据流、控制流约束关系（3.1小节中提及）。 为了验证服务行为的正确性和可靠性，必须充分的测试所有服务行为相关的数据流和控制流约束信息。每个从服务组合流程开始到结束的路径称为一条测试序列，在给定的行为模型基础上，定义各

种覆盖准则，开发相应的遍历算法，即可获取所需测试序列以及程序执行该条序列的数据约束，可通过约束求解获取该测试序列对应的测试数据，{55%：最后将数据进行封装生成测试用例，其过程如图3-11所示。}

图3-11 测试用例生成过程

因此我们重点考虑如下三个问题：

如何从给定行为模型中生成测试序列；

{63%：如何根据测试序列求解相对应的测试数据；}

{51%：如何将测试数据与测试序列结合，生成可执行的测试用例。}

行为模型驱动的测试序列生成

行为模型建立完毕后，考虑如何从 Web服务行为模型中依据覆盖算法导出目标测试序列，{49%：本文将 Web行为模型中从开始节点到结束节点的一条路径一个合规测试序列，} 对于不同测试角度而言，关注的测试序列不同，因此我们提出以下的覆盖准则[64]来决定模型中哪些元素需要进行测试。

Request-Node Coverage（请求节点覆盖）：{48%：要求设计足够多的测试序列，使得Web服务行为模型中的每个请求节点至少被覆盖一次；} 针对图3-8的行为模型，则应生成符合请求节点覆盖准则的测试序列需覆盖login与feeCalculate节点。

Response-Node Coverage（响应节点覆盖）：{48%：要求设计足够多的测试序列，使得Web服务行为模型中的每个响应节点至少被覆盖一次；} 针对图3-8的行为模型，则应生成符合响应节点覆盖准则的测试序列需覆盖loginResponse_succ、loginResponse_fail、feeCalculateResponse_succ与feeCalculateResponse_fail节点。

Edge Coverage (边覆盖)：{48%：要求设计足够多的测试序列，使得Web服务行为模型中的每条边至少被覆盖一次；} 针对图3-8行为模型，则应生成符合边覆盖准则的测试序列需覆盖 e0、e1、e2、e3、e4、e5、e6、e7、e8、e9、e10及 e11边。

State Coverage（状态覆盖）：该覆盖准则结合前三条覆盖准则一起使用，前面所述三条覆盖准则，并未考虑覆盖到该消息节点或执行边时的服务状态，{45%：状态覆盖要求设计足够多的测试用例，使得覆盖 Web服务行为模型图目标元素的同时考虑其内部状态的覆盖。} 例如针对响应节点覆盖准则，当覆盖目标为 loginResponse_ fail节点时（入库操作调用失败），有多种输入不合法的情况导致该响应节点的执行，但这种内部状态在响应节点覆盖准则中并未考虑。

本课题集成开源模型测试工具 GraphWalker[12]实现合规测试序列的自动生成，GraphWalker提供了一种图模型建模语言用以描述图模型（ 第2.1节提及）以及多种图模型遍历策略。 下面本文将介绍所使用的图模型遍历策略以及如何将定义的覆盖准则转换为相对应的遍历策略。

GraphWalker遍历策略由路径生成器(Path Generators)与停止条件(Stop Conditions)组成，如图3-12所示。 其中路径生成器是一种决定如何遍历模型的算法，生成

器提供了随机算法、起始点算法及最短路径算法； 停止条件决定何时停止该遍历过程（如模型元素覆盖率达到设定比例或特定元素被覆盖）， 停止条件有节点覆盖比例、边覆盖比例、特定节点覆盖、特定边覆盖、执行时间等。

图3-12 GraphWalker遍历策略

本文使用起始点算法及特定节点覆盖与特定边覆盖两种停止条件进行行为模型的遍历，当覆盖到特定节点或边即停止该遍历过程。 特定节点覆盖算法表达式为：a_star(reached_vertex(vertex name))； 特定边覆盖算法表达式为：a_star(reached_edge(edge name))。

起始点算法将从模型起始节点（ Start）进行遍历，生成覆盖特定节点（ vertex name）或边（ edge name）的最短路径， {45%：如图3-13 a)所示，为目标模型示例，使用起始节点算法后，生成的} 覆盖节点 v_ Browse及边 e_ Exit的执行序列如图3-13 b)所示。

a) 目标模型示例 b) 起始点算法覆盖目标节点与边的序列

图3-13 起始点算法生成目标路径示例

确定了使用的图模型遍历策略后，考虑如何将定义的覆盖准则转换为相对应的遍历策略以生成满足覆盖准则的测试序列。 因此针对本文定义的覆盖准则，需要进行如下三步转换：

1、将生成的行为模型转换为符合GraphWalker语法的graphml文件： 根据2.1小节可知GraphWalker提供了一种图模型建模语言（ Graph Model Language）用以描述图模型，后续遍历策略也基于该描述语言，因此需要使用该建模语言描述目标 Web服务行为模型。其语法如图2-4所示，需要将 Web服务行为模型中的节点映射为 graphml文件中 node标签， 将节点编号映射为 node标签 id属性内容，节点名称映射为 NodeLabel标签内容； 将Web服务行为模型中的边映射为edge标签，边的编号映射为edge标签id属性内容，边的起始点编号映射为source属性内容； {43%：边的终端点编号映射为target属性内容，边的名称映射为EdgeLabel标签内容。}

{45%：通过上述过程，可以将Web服务行为模型转换为符合GraphWalker语法的graphml文件。} 如图3-14示例演示了这一转换过程（由于关注转换过程，所以此处忽略了Web服务行为模型的其他属性）。

图3- 14 Web服务行为模型转换

{41%：将Web服务行为模型转换为graphml文件过程可总结为图3-15所示算法。} 该算法由三个主要步骤组成：

初始化（1-2）： 生成graphml文件，解析Web服务行为模型，获取其全部节点（NodeSet）与边（EdgeSet））集合。

行为模型节点转换（3-6）： 针对节点集合中的每个节点n，在graphml文件中插入Node标签，根据节点属性，填充标签内容及其属性内容。

行为模型边转换（7-10）： 针对节点集合中的每条边e，在graphml文件中插入Edge标签，根据边属性，填充标签内容及其属性内容。

图3-15 Web服务行为模型转换算法

2、将覆盖准则转换为符合起始点算法的命令： 将Web服务行为模型转换转换为graphml文件后，考虑如何使用起始点算法描述相应的覆盖准则，以生成测试序列。 本文定义了如下转换规则，使得起始点算法可表达相应的覆盖准则：

转换规则1（R1）： 针对请求节点覆盖准则，遍历Web行为模型获取请求类型节点集合，针对该集合中的元素，采用特定节点覆盖停止条件生成覆盖该节点的序列。 其覆盖算法表达式为： a_star(reached_vertex(vertex name))，其中vertex name为Web行为模型中的类型为Req的节点名称。

转换规则2（R2）： 针对响应节点覆盖准则，遍历Web行为模型获取响应类型节点集合，针对该集合中的元素，采用特定节点覆盖停止条件生成覆盖该节点的序列。 其覆盖算法表达式为： a_star(reached_vertex(vertex name))，其中vertex name为Web行为模型中的类型为Res的节点名称。

转换规则3（R3）： 针对边覆盖准则，遍历Web行为模型获取边集合，针对该集合中的元素，采用特定边覆盖停止条件生成覆盖，其覆盖算法表达式为：a_star(reached_edge(edge name))，其中edge name为Web行为模型中的边。

3、去除冗余序列： 使用上述转换规则生成的测试序列存在序列冗余问题，例如对于图3-8所示行为模型，采用响应节点覆盖准则生成对feeCalculateResponse_succ节点的测试序列为：StrateOInitellogine2loginResponse_succe5feeCalculatee7feeCalculateResponse_succ。包含了针对loginResponse_succ节点生成的测试序列StrateOInitellogine2loginResponse_succ。 因此需要进一步考虑去除冗余序列。

通过上述描述，针对服务行为模型生成合规测试序列的过程可总结为图3-16所示算法。

图3-16 合规测试序列生成算法

该算法由四个主要步骤组成：

初始化（1-2）： 定义初始测试序列集合initTss=及测试序列集合tss=、选取覆盖准则；

覆盖元素选取（3-12）： 针对不同覆盖准则，选取不同覆盖元素集合eleCoverSet。{41%：针对请求节点覆盖准则，覆盖元素为Web行为模型中节点类型为Req的节点；} {41%：针对响应节点覆盖准则，覆盖元素为Web行为模型中节点类型为Res的节点；} 针对边覆盖准则，覆盖元素为Web行为模型中所有边； 以图3-8为例，若选取响应节点覆盖准则，则有eleCoverSet = {v3，v4，v6，v7}。

初始测试序列集合生成（13-16）： 针对覆盖元素集合中的每个元素，集成GraphWalker生成对应测试序列，得到初始测试序列集合。 以图3-8为例，针对集合eleCoverSet={v3，v4，v6，v7}生成的初始测试序列集合initTss为：

StrateOInitellogine2loginResponse_succ

StrateOInitellogine3loginResponse_fail

StrateOInitellogine2loginResponse_succe5feeCalculatee7feeCalculateResponse_succ

StrateOInitellogine2loginResponse_succe5feeCalculatee8feeCalculateResponse_fai

1

测试序列精简（17-26）： {46%：针对生成的初始测试序列集合进行冗余测试序列删减，得到最终合规测试序列集。} 以步骤三生成的初始测试序列集合 initTss为例，选取集合中最长测试序列 Strate0 Inite1 logine2 loginResponse_ succe5 feeCalculatee7 feeCalculateResponse_ succ， 该序列覆盖 eleCoverSet中 v3，v6元素，将该序列加入测试序列集合 tss中， eleCoverSet= eleCoverSet-{ v3，v6}={ v4， v7}。 由于 eleCoverSet不为空，因此选取初始集合中次长序列 Strate0 Inite1 logine2 loginResponse_ succe5 feeCalculatee8 feeCalculateResponse_ fail， 该序列覆盖 eleCoverSet中 v6元素，将该序列加入测试序列集合 tss中， eleCoverSet= eleCoverSet-{ v6}={ v3}。 由于eleCoverSet不为空，因此选取初始集合中Strate0Inite1logine2loginResponse_succ序列，由于该序列并不覆盖eleCoverSet中元素，因此不加入tss中。 由于 eleCoverSet不为空，因此选取初始集合中 Strate0 Inite1 logine3 loginResponse_ fail序列，该序列覆盖 eleCoverSet中 v3元素， eleCoverSet= eleCoverSet-{ v3}=，将该序列加入测试序列集合 tss中。 由于eleCoverSet为空，因此结束精简。 最终得到的测试序列tss为：

Strate0Inite1logine3loginResponse_fail

Strate0Inite1logine2loginResponse_succe5feeCalculatee7feeCalculateResponse_succ

Strate0Inite1logine2loginResponse_succe5feeCalculatee8feeCalculateResponse_fail

{43%：根据以上四个步骤，可针对图3-8行为模型生成满足响应节点覆盖准则的合规测试序列。}

{59%：前文描述了预期情况的测试序列生成过程。} 然而，同样重要的是测试违反约束规定的服务调用情况，在这种情况下，服务不能按预期工作。 {53%：我们将这种测试序列称为冲突测试序列。} {54%：冲突测试序列生成主要生成违反序列约束的测试序列。} 针对重复调用约束为false的操作，生成重复调用该操作的序列，针对有顺序约束的操作，生成单独调用该操作序列。 图3-17描述了冲突测试序列的生成算法。

图3-17 冲突测试序列生成算法

该算法由三个主要步骤组成：

集合初始化（1-2）： 初始化冲突测试序列集合cTss=，解析行为模型获取请求节点集合ReqNodeSet。 针对图3-8有，ReqNodeSet={v2，v5}。

重复调用约束冲突序列生成（4-7）： 针对请求节点集合中的每个元素，获取其重复调用约束，若该约束为false，生成请求节点执行成功后重复调用该节点的序列。 根据图3-9可知，v2节点重复调用约束为false，因此生成Strate0Inite1logine2loginResponse_succeflogin序列。 v5节点重复调用约束为false，因此生成Strate0Inite1logine2loginResponse_succe5feeCalculatee7feeCalculateResponse_succeffeeCalculate序列，ef代表错误转移边。

顺序约束冲突序列生成（8-11）： 针对请求节点集合中的每个元素，获取其顺序约束，若该约束为空，则该请求节点对应操作可直接调用。 否则，该节点需依据按一定顺序的正确执行，生成直接调用该节点的错误序列。 根据图3-9可知，v2节点序列约束为空，无需生成冲突序列，v5节点序列约束不为空，因此生成StarteOIniteffeeCalculate序列。

根据以上四个步骤，可针对图3-8行为模型生成冲突测试序列。

{67%：基于约束求解策略的测试数据自动生成}

本节介绍基于约束求解的测试数据自动生成，包括约束条件表达式的提取、约束求解工具Z3自动生成可行解。

{45%：在前面的论述中，我们知道测试用例生成包括测试序列的生成、测试数据的生成。}上一小节，已经根据不同覆盖策略生成了测试序列，生成测试序列后，考虑如何根据测试序列，生成满足该序列的测试数据。

传统测试方法中，测试数据的获取主要依靠人工分析来完成，不仅繁琐而且耗时，覆盖率不高。 {46%：为了保证生成测试用例的完备与有效性并且提高测试质量，本文针对每条测试序列自动生成测试数据。}

本文将已有的测试序列，进一步采用约束求解技术（Constraint Solver），在测试序列基础上产生所需测试数据。 {51%：在约束求解步骤中综合使用线性约束求解策略与迭代求解策略，线性约束求解策略求解出所有的解，} 然后用迭代求解的策略选出满足约束的解，从而得出满足每条测试序列的测试数据。 该方法处理流程如图3-18所示。

图3-18 测试数据生成流程

{44%：约束求解包括如下两个步骤，一是提取约束条件，即从行为模型及决策表中提取出对应序列的所有约束条件表达式；} {54%：二是求解约束，即采用正确的算法和策略，求出约束条件表达式的可行解。}

{47%：下面重点介绍测试数据生成流程中的关键技术：} 调用约束求解工具Z3自动生成可行解、约束条件表达式的提取与转换。

调用约束求解工具Z3自动生成可行解

Z3提供了很多的底层接口（SMT-LIB、C/C++、.NET、JAVA）来实现约束条件表达式的求解，并且拥有自己的语法格式。 本文使用SMT-LIB语言编写Z3求解脚本(ScriptZ3.txt)，使用z3 ScriptZ3.txt命令调用脚本获取符合约束的解。

{45%：Z3 SMT-LIB求解脚本的输入格式有变量定义、运算及模型求解三个部分。}

变量定义部分使用declare-const命令定义需要求解的变量： Z3内置整数(Int)、实数(Real)、布尔数（Bool）、字符串（String)及常量支持，declare-const命令语法为：declare-const 变量名 变量类型。

运算部分使用assert命令添加约束到Z3内部堆栈，用于后续求解： 巴科斯范式表示assert命令规则assert [term]如下所示：

模型求解部分使用get-model命令获取约束解，也可以使用get-value (var)命令获取特定变量var的求解值。

{45%：如图3-19所示为Z3求解脚本示例及其求解结果，约束要求为：} (a+b)[=5 a]=2 d=hello，Z3判定约束表达式可求解后（使用chack-sat命令判断）即输出所求变量可行解： a=2; b=0; c=false; d=hello。

图3-19 Z3求解脚本示例及其求解结果

约束条件表达式的提取

接下来考虑如何从测试序列中获取序列执行约束并将约束转换为符合Z3 SMT格式的脚本。如图3-18所示，约束条件表达式集合需依据测试序列，从行为模型与服务提供商提供的决策表中提取。 由于决策表是处理基于约束的事件序列的有效机制[41]，本文使用决策表定义操作对于不同输入数据的响应。

决策表（DT）为一个三元组DT=[C， E， R]。 C表示可以被赋值为true或false的约束条件集合； E表示响应事件的集合； {78%：R表示任何一个条件组合的特定取值及其相应执行事件。} {50%：待测服务的一个操作对应一个决策表。}

表3-6为例3-1入库操作（login）对应的包含5个约束、2个事件、4个规则的决策表。 为保证自动化求解决策表中约束条件的集合，决策表中的约束条件使用Z3求解器能够解释的assert命令进行表达。 由于测试序列中存在多次调用同一操作的情况，使用? 标识操作调用次数，根据调用的次数赋值，用以区分操作参数。

根据表3-6可知，当且仅当 License为以 BJA- BJY开头，5位数字结尾的字符串且loginTime属于0到24范围的整数时， login操作执行并反馈 loginResponse_ succ响应；其余情况均响应loginResponse_fail。

表3-6 login操作决策表

下面根据例3-1及表3-6，描述一个测试数据的生成过程。 针对测试序列StrateOInitellogine3loginResponse_fail，遍历该序列，对该序列中的节点进行如下处理：

请求节点： 对于节点类型为 Req的请求节点，从 Web行为模型中获取该节点约束，解析其输入参数名称及类型， 转换为符合 Z3脚本的 declare- const命令，进行变量定义。 针对login节点，解析Web行为模型中login节点约束，其输入参数为String类型的License及Int类型的loginTime。 转换为符合declare-const变量定义脚本为：

```
(declare-const login_1_License String)

(declare-const login_1_loginTime Int)
```

响应节点： 对于节点类型为 Res的响应节点，解析该节点对应的操作决策表，选取执行事件与响应节点名称相同的规则， 获取其规则中约束条件为真的集合，转换为符合Z3 assert命令的脚本。 针对loginResponse_fail节点，解析表3-6，可选取规则1，2或3。假设选取规则1，解析规则1中约束条件为真的集合，其Z3 assert命令的脚本为：

```
( assert(= false( str. in. re login_1_ License( re.++( re.++ ( re.++( re.++( re.++( re.++( str. to. re” BJ”)( re. range” A”” Y”))( re. range”0”” 9”) )( re. range”0”” 9”))( re. range”0”” 9”))( re. range”0”” 9”))( re. range”0”” 9”))( re. range”0”” 9”)))))

(assert (]= login_1_loginTime 0))
```

(assert ([= login_1_loginTime 24))

序列遍历结束后，生成的 Z3求解脚本包括了待求解变量定及运算部分， 还需向脚本中添加模型求解命令，判断是否有可行解并获取最后所求解数据。 {45%：针对该测试序列生成的测试数据求解脚本，及其求解结果如图3-20所示。}

图3-20 测试数据求解脚本及结果示例

图3-21 测试数据生成算法

本文在3.3.1小节覆盖准则中定义了状态覆盖（ State coverage）准则， 若使用该覆盖准则，则需针对所有符合要求的规则生成测试数据，如针对 loginResponse_ fail节点， 解析表3-6，需对规则1，2，3分别生成测试数据。 即针对测试序列StrateOInitellogine3loginResponse_fail，应生成三个测试用例。 假设一个测试序列由 n个响应节点组成，每个响应节点 i具有 mi个对应规则，则针对该序列， 应该生成小于等于 m1 m2 mn个测试数据（存在规则组合无可行解的情况）。

图3-21描述了测试数据自动生成算法。 该算法首先遍历测试序列，获取全部类型为请求节点的节点集合（ReqNodeSet）与类型为响应节点的节点集合（ResNodeSet）。 针对响应节点集合，从决策表中获取执行相关响应的所有可能的规则组合集合（RCSet）。 若使用状态覆盖准则（State coverage=true），则针对每个规则组合进行测试序列数据求解；若未使用该覆盖准则，则针对该序列，求出一种规则组合下的可行解即可。

{63%：基于测试序列及测试数据的测试用例生成}

{46%：生成测试数据后，需要考虑如何将测试数据与测试序列结合，生成可执行的测试用例。}

本文使用XML Format定义测试用例，其语法如图3-22所示。 其中根节点为[Sequence]标签，用来标识测试用例； [Operation]标签标识被调用操作， name属性代表被调用操作名称， id属性代表操作被调用次数，[Sequence]标签中可包含多个[Operation]标签， [Operation]标签的前后顺序即为服务调用的先后顺序；[Operation]标签内包含调用该服务需要发送的SOAP消息，[Body]标签内包含消息所需测试数据。

图3-22 测试用例框架

{43%：定义完测试用例格式后，本文考虑如何将测试数据与测试用例进行结合。} 在前面的技术框架图论述中本文使用 Z3求解器求解出的测试数据填充到 EX- WSDL分析阶段生成的 SOAP消息中， {54%：将填充后的消息与生成的测试序列合成可执行的测试用例。} 如图3-23所示，测试用例从数据层、消息层、序列层三个层次进行生成：

基于约束求解生成测试数据；

将测试数据填充到EX-WSDL分析阶段生成的相应SOAP消息框架中，生成单个操作测试消息；

{57%：按照测试序列，将测试消息进行组合后，生成最终的测试用例。}

图3-23 测试用例生成框架

针对测试序列StrateOInitellogine3loginResponse_fail与图3-20生成的可行解，可生成

最后的测试用例如图3-24所示。

图3-24 测试用例示例

测试执行与结果判定

本文集成SoapUI开源工具模拟客户端，并通过HTTP进行数据传输，将SOAP请求发送到该服务所在的服务器进行测试用例的执行。 测试与监控服务的运行以及对服务操作的调用是否符合约束条件进行判断，针对执行错误的测试用例，尝试定位违反约束的类型。 因此本文定义了约束校验优先级，限定了服务调用失败后，约束判定的先后顺序。

如表3-7所示，约束校验优先级分为P1-P7，编号越小，越优先进行判断。

表3-7约束校验优先级

{58%：测试执行及结果判定算法如图3-25所示。}

在上述算法中，首先解析测试用例tc中的调用请求消息集合（ReqMSet），针对该集合中的每个请求消息进行执行判定。 根据约束校验优先级进行如下处理：

如果请求消息不符合调用操作的 XML结构定义（ XSD）停止测试用例执行，记录该测试用例违反参数范围（ paraRestriction Constraint）约束（5-7）， 否则继续执行；

获取执行后的响应消息，判断该消息是否符合调用操作的XML结构定义，若满足则对下一请求消息执行步骤1； 否则执行步骤3-8进一步判断；

判断响应消息是否存在找不到服务提示，若存在服务可能存在由于更新或停止等问题导致操作被移除， {58%：停止测试用例执行，记录该测试用例违反时效约束（12-15）；} 否则进一步判断；

判断客户端IP地址是否符合调用操作区域约束，若不在限定区域内，停止测试用例执行，记录该测试用例违反区域约束（16-19）； 否则进一步判断；

如果调用操作存在重复调用约束，判断已经执行的操作序列是否存在重复调用，若存在，{41%：则停止测试用例执行，记录该测试用例违反重复调用约束（20-23）；} 否则进一步判断；

判断调用该操作前的执行操作序列是否满足调用操作的顺序约束，若违反该约束，{46%：停止测试用例执行，记录该测试用例违反顺序约束（24-27）；} 否则进一步判断；

判断请求消息中的参数关系是否满足调用操作的参数关系约束，若违反该约束， {46%：停止测试用例执行，记录该测试用例违反顺序约束（28-31）；} 否则进一步判断；

图3-25 测试结果判定算法

判断调用操作是否存在调用约束，若存在，停止测试用例执行，记录该测试用例可能违反调用约束（32-35）； 否则进一步判断；

若并未违反本文定义的约束，则提示该测试用例可能违反未知约束（36）。

{49%：行为模型驱动的服务组合程序测试用例生成工具MDGen设计与实现}

本章介绍行为模型驱动的服务组合程序测试用例生成技术支持工具MDGen的设计与实现，包括需求分析、工具设计与实现，最后用一个实例进行演示。

需求分析

为了提高行为模型驱动的服务组合程序测试用例生成技术的自动化程度，本文开发了相应支持工具MDGen。 {44%：MDGen支持EX-WSDL解析、行为模型的生成、测试序列生成、测试用例生成、执行和测试结果验证。} {50%：采用UML用例图对MDGen进行需求分析，如图4-1所示。}

图4-1 MDGen用例图

{44%：下面将对图4.1进行各个用例的具体描述。}

待测程序选择：

用户提供待测Web服务的EX-WSDL访问地址，系统根据地址获取EX-WSDL文档，并创建一系列所需工作目录。 包括如下两个子用例：

待测程序导入： 用于待测服务的 EX- WSDL文档的导入，用户可提供该服务 EX- WSDL的 URI或者本地文档的绝对路径， 工具将获取的文档导入到特定目录，以备后续相关操作。

工作目录生成： 工具根据用户选定的待测服务，生成后续操作需要的工作目录。

行为模型生成：

解析EX-WSDL并将解析结果转换为服务的行为模型。 包含如下三个子用例：

扩展后WSDL解析： 用于解析EX-WSDL，识别出该服务的约束描述、提供的操作、操作的输入/输出参数列表、操作调用消息框架以及操作的约束描述，存储解析结果。

服务约束提取： 通过解析服务的约束描述及操作的约束描述，获得服务及操作具有的特定约束。

行为模型生成： 针对提取的服务及操作约束，根据设计的行为模型生成算法，建立基于事件序列图的 Web服务行为模型，使用 Graphviz进行行为模型的可视化， 并将行为模型转换为符合 GraphWalker的图模型语言，方便后续测试用例生成。

测试用例生成：

{49%：生成行为模型后，根据覆盖准则遍历行为模型生成测试用例。} 包含如下三个子用例：

测试序列生成： {48%：根据生成的行为模型，基于一定覆盖准则，获得测试序列。}

测试数据生成： {41%：针对每条测试路径中的操作以及操作相关变量的约束条件生成对应的测试数据。}

测试用例生成： {43%：根据服务操作调用框架，将测试数据与测试序列相结合，生成可执行的测试用例。}

测试执行：

{50%：控制测试执行，从测试用例集检索测试用例、监视测试运行、搜集测试结果最终生成测试报告。} 包含如下三个子用例：

测试用例导入： 导入生成的测试用例。

测试用例执行： 在待测程序上运行测试用例，收集和记录服务执行结果，依据服务行为模型，检查对服务操作的调用，报告不满足约束的服务调用情形。

测试结果统计： {47%：对测试结果进行统计，给出执行通过与失败的测试信息，并生成测试报告。}

测试报告查看：

{45%：测试者可查看测试报告，获取更加全面的测试信息，包含执行了哪些测试用例、测试用例执行结果等信息。}

MDGen设计与实现

针对需求分析，本节详细介绍MDGen工具的架构、各组件的设计及实现。

系统架构

图4-2描述了MDGen工具的系统架构。 选用橙色矩形框表示工具的基本组件，包括EX-WSDL解析组件，测试序列生成组件，测试用例生成组件及测试用例执行组件共四个组件。每个组件由多个模块构成，其中测试序列生成组件由行为模型生成及测试序列生成模块构成、测试用例生成组件由测试数据生成及测试脚本生成模块构成。 各个组件的功能设计描述如下：

EX-WSDL解析： 负责对待测程序的扩展后WSDL文件解析，生成相应行为约束文档（ParseResult），包括三个子模块。

EX-WSDL解析器： 该模块根据用户输入的EX-WSDL文件路径，读入待解析的EX-WSDL文件，并进行解析。

XML文件读/写器： 负责生成待测服务操作的输入输出XML结构定义文件（XSD），为后续验证做准备。 通过DOM4J技术读入EX-WSDL文件，将其解析为DOM树形结构的Document对象，对其中的schema节点进行查找，完成XSD文件的输出。

SOAP消息生成器： 集成soapUI工具，调用其API生成服务操作SOAP消息框架，便于后续测试数据填充。

图4-2工具系统架构图

测试序列生成组件： 主要负责解析扩展WSDL解析器生成的行为约束文档，转换为行为模型图（WSBM）； {53%：进一步对模型进行遍历，生成满足一定覆盖准则的测试路径。}该组件包含两部分：

行为模型生成组件

行为模型生成器： 该模块通过解析EX-WSDL解析组件生成的行为约束文档，根据行为模型生成算法，生成对应的行为模型（WSBM）。

模型可视化： 该模块通过集成Graphviz，可视化展示行为模型。

行为模型转换器： 根据模型转换算法，将行为模型转换为GraphWalker支持的图模型文档（graphml），方便后续测试序列生成。

测试序列生成组件

合规测试序列生成器： {45%：该模块根据用户选择覆盖准则，遍历graphml文档，生成满足特定覆盖准则的合规测试序列。}

冲突测试序列生成器： {41%：该模块根据冲突测试序列生成算法，遍历行为模型（WSBM），生成冲突测试序列。}

测试用例生成组件： 该组件主要依据测试序列求解出执行该序列的测试数据，并将测试序列与测试数据进行合成生成测试用例，该组件包含两个部分：

测试数据生成组件： 该组件主要负责提取测试序列约束条件表达式，集成开源工具Z3对提取出的条件表达式进行求解，生成满足约束条件的可行解。

约束提取器： 该模块主要针对每条测试序列，提取出执行该序列的约束条件，将其转换为符合Z3脚本语法的约束条件表达式。

约束求解器： {41%：该模块集成开源工具Z3，对提取出的约束条件表达式进行求解，生成满足约束条件的可行解。}

测试脚本生成组件： {45%：该组件根据服务操作调用框架，将测试数据与测试序列相结合，生成可执行的测试用例。}

数据填充器： 该模块主要将测试数据填充到相应的SOAP消息框架中（消息框架已由解析器解析得到），生成SOAP消息；

脚本生成器： 该模块根据测试序列中的操作调用顺序，将填充的SOAP消息进行合成，生成可执行的测试用例。

测试用例执行组件： {47%：该组件负责执行测试用例并评估执行结果、生成测试报告，由如下三个模块组成：}

用例执行器： 该模块依次读取测试用例生成组件生成的测试用例并解析、集成soapUI模拟客户端调用Web服务进行测试用例执行，保存执行过程中的输出结果。

结果评估器： 该模块监控用例执行过程，截取执行器执行过程中的请求响应信息、将截取信息与解析器组件生成的 XSD文档进行校验， 根据定义的校验准则，生成测试评估结果。

结果统计器： 该模块获取结果评估器生成的评估结果，对其进行统计分析并生成测试报告， {45%：包括生成测试用例总个数、违反服务行为约束测试用例个数及违反约束情况。}

工具实现

下面讨论支持工具MDGen的实现。 行为模型可视化通过集成Graphviz实现，测试数据生成主要通过集成Z3约束求解器实现，测试用例执行通过集成soapUI模拟客户端实现。 MDGen核心功能的类分为5大模块：

扩展WSDL解析

主要对输入的 EX- WSDL使用 soapUI及 DOM4 J技术进行解析， {42%：获取该服务的约束描述、提供的操作、操作的输入输出参数列表、操作调用消息框架以及操作的约束描述，} 存储约束解析结果（约束文档）。

ReadWsdl类： 用于解析EX-WSDL文件，获取服务提供的操作、操作的输入/输出参数列表、操作调用消息框架以及操作的约束描述，存储约束解析结果。

SampleSoapBuilder类： 用于解析操作的输入输出格式要求，输入操作对象，解析EX-WSDL文档获取该对象的输入输出参数格式、名称、类型及其约束。

XmlInputFormat类： {54%：用于存储操作输入参数名称、类型及其约束。}

行为模型生成

用于根据EX-WSDL文档解析获得的约束文档生成Web服务行为模型图。

Graph类： {48%：行为模型定义类，用于描述Web服务行为模型。}

Coverter类： 完成从约束文档到Web行为模型的转换，生成Web行为模型； 将Web行为模型转换为GraphWalker支持的graphml文件。

ModelVisualization类： 完成行为模型可视化。

测试序列生成

{63%：根据定义的覆盖准则，遍历Web行为模型生成测试序列。}

TeatSequence类： {51%：测试序列定义类，该类定义测试序列。}

CoverageCriteria类： {55%：覆盖准则定义类，该类用于定义覆盖准则。}

GetInitialTestSequence类： 根据用户选取的覆盖准则，遍历Coverter类生成的graphml文档，生成满足特定覆盖准则的测试序列。

测试用例生成

针对特定测试序列，从 Web行为模型与决策表中获取该序列执行约束，使用约束求解器求解出满足约束数据后， {61%：将测试数据与测试序列合成测试用例。}

ParseDT类： {45%：决策表解析类，解析决策表，获取并存储响应事件对应的规则。}

TestDateG类： 测试数据生成类，针对特定测试序列，从 Web行为模型与决策表中获取该序列执行约束， 将约束转换为 Z3求解脚本，调用 InvokeZ3类将求解结果进行解析，保存求解数据。

InvokeZ3类： {41%：约束求解器调用类，该类用于调用Z3执行约束求解脚本，并保存满足约束的求解结果。}

DataToCase类： 测试用例合成类，该类将测试数据按照ReadWsdl类解析出的操作调用消息框架转换为SOAP测试脚本。

测试用例执行及结果统计

{48%：通过集成 soapUI模拟客户端实现测试用例的执行，收集和记录服务执行结果，}依据服务行为模型，检查对服务操作的调用，报告不满足约束的服务调用情形。

ScriptToFrame类： 该类用于测试用例的执行与监控。 通过解析DataToCase类生成的测试用例，调用SendWsdlInterface类执行解析出的SOAP消息，截取SendWsdlInterface执行结果。 将SOAP消息、截取的执行信息与ReadWsdl类解析出的XSD文档进行校验，根据定义的校验准则，生成测试评估结果。

SendWsdlInterface类： {44%：通过集成soapUI模拟客户端发送SOAP消息，获取消息执行结果。}

ResultReport类： 外观类。 用于解析ScriptToFrame生成的测试结果并可视化展示。

系统演示

MDGen工具主要由四个部分组成： {41%：菜单栏、文件选择区域、具体功能区域和日志区域。} 其中，菜单部分提供工具的重启与退出、Tomcat的运行、帮助三个菜单项；文件选择区域提供选择待测Web服务EX-WSDL地址、导入相关联决策表及清除缓存三个功能；具体功能区域是工具的主要操作点，包括EX-WSDL解析、行为模型生成、测试序列生成、测试用例生成、测试用例执行分析五个部分； 日志区域提供工具使用过程与用户操作行为相关的日志，例如EX-WSDL解析过程、测试用例执行过程输出的日志。

采用例3-1实例来演示MDGen工具的使用，包括EX-WSDL解析、行为模型生成、测试序列生成、测试用例生成、测试用例执行分析部分。

EX-WSDL解析过程演示

在解析EX-WSDL文件前，需要输入待解析文件URL或地址，同时需要点击[Browse Related Excel]按钮导入与测试服务相关的决策表。 点击[Chose]按钮后，系统创建针对选择待测服务的工作目录，进入解析界面[Parse WSDL]，可对目标EX-WSDL文件进行解析。

首先在[Parse WSDL]框内点击[Parse]按钮，系统针对选择的EX-WSDL文档进行解析。解析结果显示在[Reult]框中，解析结果包括服务名称、提供操作、操作名称及操作的具体的消息访问接口； 解析过程的输出信息显示在[Console]框中； 操作序列约束内容显示在[Model Set]框中，用户可通过下拉框选择不同操作， 在[Sequence Constraint]框中查看该操作的序列约束（ preOP）、设置操作消息之间的 关联（ From To）并点击[Add]按钮将其关系添加进行为模型中。 图4-3展示该实例的EX-WSDL解析情况。

图4-3扩展WSDL解析界面

若服务时效约束时间小于目前系统时间，则在解析过程中，系统会提示用户该服务可能存在由于接口变动导致操作不可用问题， 如图4-4所示。

图4-4 服务时效约束检测

行为模型生成演示

在[Parse WSDL]标签页点击[Generate Model]按钮，系统按照行为模型生成算法，生成相关行为模型，进入[Generate TestSequences]界面。 为了更直观的展示行为模型，系统提供一个[Model View]界面框可视化的显示行为模型，用户点击[ShowPic]按钮，即可显示生成的行为模型图，图4-5展示了该实例行为模型可视化预览界面。

图4-5 行为模型生成预览界面

测试序列生成演示

系统[Generate TestSequences]标签页支持测试序列生成，提供了覆盖方法选择、测试序列生成两个主要功能。 其中[Coverage Criteria]框支持Request-Node（请求节点）、Response-Node（响应节点）、Edge（边）及State（状态）覆盖准则选择。 用户选定覆盖准则后，点击[Generation]按钮，系统依据测试序列生成算法，生成满足覆盖准则的测试序列。 [Test Sequences]框用来显示生成的测试序列，包括序列编号（ Id）、序列类型（ Type）、序列内容（ Sequence）及覆盖元素（ Coverd Elements），使得用户能够更直观的了解到生成的测试序列，也可配合行为模型图进行校验。 图4-6展示了针对例3-1，使用边覆盖准则生成的测试序列。

图4-6 测试序列生成界面

测试用例生成演示

在[Generate TestSequences]标签页中点击[TestCase Generate]按钮，系统将调用Z3约束求解器，针对每个测试序列生成测试用例。 此时进入[Generate TestCase]标签页。 [Statistics]框提供测试用例统计功能，显示针对每条测试序列（Id），共生成多少测试用例（Nmuber）。 [TC View]提供测试用例查看功能，使用下拉框选择所要查看的测试用例，[Test Case]框将会给出用户所要查看的测试用例内容。 {45%：[Test Execution]框提供测试用例执行功能，点击[Execution]按钮可进行测试用例执行。} 如图4-7所示为测试用例生成界面。

图4-7测试用例生成界面

测试用例执行分析演示

{44%：通过以上步骤，得到行为模型和测试用例集合后可以进行测试执行。} 通过点击[Generate TestCase]标签页的[Execution]按钮进行测试用例执行，系统将会自动执行测试用例并将测试结果保存在文件中， 系统监控测试执行并将执行错误信息输出到[Console]，包括测试用例名称及错误原因，如图4-8， {51%：所示为系统执行测试用例过程的输出信息。}

图4-8 测试执行过程日志

{58%：完成测试用例的执行后，系统自动对测试结果进行统计分析。} {44%：其中，[Statistic Report]显示了全部测试用例、合规测试用例及冲突测试用例个数与所占比例。} [Detailed Information]显示了测试用例违反约束的情况，包括违反各个约束的测试用例个数及其所占比例。

图4-9 测试结果分析界面

{41%：本实例中，共7违反参数范围约束的测试用例、36个违反顺序约束的测试用} 例、37个违反重复调用约束的测试用例及2个违反参数关系约束的测试用例， 分别占全部违反约束测试用例的8.54%、43.90%、45.12%及2.44%。 通过点击[Detailed Information]表格内容进入结果详细分析界面[Detail]。 [Detail]提供了详细的结果查看功能，点击表格内容后，[Detail]显示违反目标约束的测试用例列表， 使用下拉菜单选择查看的测试用例后，系统显示该测试用例执行后的输出结果以及错误信息。 图4-9显示了测试分析结果。

小结

本章详细讨论了MDGen工具的设计与实现。 {46%：该工具支持EX-WSDL解析、行为模型的生成、测试序列生成、测试用例生成、执行和测试结果验证。} {55%：有助于提高所提技术的自动化程度。}

实例研究

为评估行为模型驱动的服务组合程序测试用例生成技术的有效性及支持工具的实用性，本章选择两个Web服务进行实例研究。 两个实例程序均根据真实规格说明开发。

研究问题

{59%：本章实验将围绕以下三个问题展开讨论：}

验证EX-WSDL对于本文提出的服务行为约束的表达能力以及基于EX-WSDL的服务行为模型生成技术是否能够正确生成服务行为模型；

验证行为模型驱动的服务组合程序测试用例生成技术的可行性，能够生成合规及冲突的测试用例，能否正确报告违反服务行为约束的错误调用；

{41%：针对不同覆盖准则进行评估，重点关注不同覆盖准则生成的测试用例的违反服务行为约束调用的检测情况。}

实验对象

本文实验对象共包括2个Web服务实例。 停车计费服务（记为PFC）及费用补偿服务（EXP），每个实例程序各有两个实现版本。

停车计费服务PFC

停车计费服务依据司机的车辆类型（摩托车、跑车或轿车），停车日期（工作日或周末），折扣券和停车时间计算停车费用， 提供入库（ login）及出库计费（ feeCalculate）操作，车辆入库时调用入库操作，出库时调用计费操作， 出库及计费操作输入规格说明分别如表5-1及5-2所示。

表5-1 入库操作输入规格说明

在表5-1中， License代表车辆车牌号，使用 BJA- BJY开头，五位数字结尾的字符串， LoginTime代表车辆入库时间，为0到24中的任意时刻。

表5-2 出库计费操作输入规格说明

在表5-2中，License代表出库计费车辆车牌号，应该与入库时相同； type代表车辆类型，由枚举值{0，1，2}分别代表摩托车、跑车及轿车； timeout代表车辆出库时间，停车时间的计算方法为timeout-loginTime，因此要求timeout大于等于loginTime； dayOfWeek代表停车日是否为工作日； discountCoupon表示车主是否使用优惠券。

当输入符合服务规格说明时，根据停车时间，是否享受折扣，停车单价计算出最终的停车费用。 停车单价计算方式如表5-3所示。

表5-3 停车计费单价计算规则

根据上述描述可知 PFC服务具有参数范围约束（表5-1、5-2中约束） 、参数关系约束（入库与出库操作的车牌号相同、车辆出库时间大于等于入库时间） 、序列约束（入库成功后可调用出库操作、入库成功后不可重复调用入库操作）。 本文在 PFC服务中考虑时效约束生成了 PFC2，程序实现过程中删除出库操作功能，但并未对 PFC2的 WSDL接口进行修改， 用以模仿由于服务实现修改后带来的操作不可用问题。

费用补偿服务EXP

费用补偿系统协助公司销售总监： （1）确定每个高级销售经理和销售经理因使用公司车辆产生的过度英里数而应向公司补偿的费用（ 2）处理高级销售经理、销售经理和销售主管有关机票、酒店住宿、吃饭和电话等各种类型的补偿请求。 该服务仅针对公司内部系统提供，支持高级销售经理、销售经理和销售主管的费用报销计算。 提供车辆使用费计算（calAmount）机票报销（airfareReimburse）及总金额计算（totalAmount）三个操作。其中，车辆使用费计算操作根据每个高级销售经理和销售经理因使用公司车辆产生的过度英里数计算应向公司支付的费用， 其输入规格说明如表5-4所示； 机票报销操作用以计算销售经理、销售经理和销售主管有关机票购买及其他出差事务所花费的费用，其输入规格说明如表5-5所示； 总金额计算操作调用车辆使用费计算操作及机票报销操作，计算最后应为高级销售经理、销售经理或销售主管报销的总费用， 其输入规格说明如表5-6所示。

表5-4 车辆使用费计算操作输入规格说明

其中，stafflevel代表员工等级，该操作仅支持针对高级销售经理（seniormanager）、销售经理（manager）的车辆使用费征收。 mileage表示当月实际英里数，即使用公司车辆的英里数。 当输入符合操作规格说明时，根据员工等级，公车使用英里数，计算出最终的补偿金额。 补偿金额计算方式如表5-7所示。

表5-5 机票报销操作输入规格说明

其中，stafflevel代表员工等级，该操作支持针对高级销售经理（seniormanager）、销售经理（manager）及销售主管（supervisor）的车辆使用费征收。 salesamount表示该员工当月实际销售额。 airfareamount表示当月机票费。 other表示当月其他申请报销的费用。 当输入符合操作规格说明时，根据员工等级、月销售额、机票费用及其他申请报销费用计算出报销金额。 报销金额计算方式如表5-8所示。

表5-6 总金额计算操作输入规格说明

其中，stafflevel代表员工等级； mileage表示当月实际英里数，即使用公司车辆的英里数； salesamount表示该员工当月实际销售额； airfareamount表示当月机票费；othere表示当月其他申请报销的费用。 总金额计算方式如表5-9所示。

表5-7 车辆使用费计算方法

车辆使用费对于不同等级的员工有不同的计算方法。 对于每一个高级销售经理和销售经理，每月有定量的公司车辆使用英里数限额（x公里），高级销售经理为x=4000，销售经理为x=3000。 若一个员工当月实际车辆使用英里数（y公里）没有超过限额的话，那么他不需要向公司偿付额外英里数费用； 如果超过限额的话，那么需要根据表5-7计算向公司偿付的额外英里数费用。

表5-8 报销金额计算方法

报销金额对于不同等级的员工有不同的计算方法。 如表5-8所示，当销售经理月销售额达到50，000时可以享受飞机票报销、销售主管月销售额达到80， 000时可以享受飞机票报销。 同时若当月销售额不少于100，000，任何等级的员工都能享受其他费用报销。

表5-9 总金额计算方法

根据上述描述可知 EXP服务具有参数范围约束（表5-4、5-5、5-6中约束） 、调用约束（总金额计算操作调用车辆使用费计算操作及机票报销操作完成目标功能） 及区域约束（车辆使用费计算操作及机票报销操作仅针对公司内部系统提供）。

由于存在区域约束，本文实验采用两个版本EXP服务，分别记为EXP1与EXP2，两个版本除了区域约束范围不同，其余服务实现及描述均相同。 实验环境IP地址不符合EXP1区域约束，但符合EXP2区域约束。

表5-10给出这些实例的具体信息，包括实例的基本功能描述（Basic functionality）、实例包含的约束（Contained constraints）。

表5-10实例程序基本信息

实验设计

按照如下步骤使用MDGen工具对待测程序进行实例研究：

服务行为分析与WSDL扩展： {42%：通过扩展WSDL描述服务行为约束并验证带有行为约束信息的EX-WSDL是否能够正确发布。}

Web服务行为模型生成： {51%：使用MDGen工具解析实验程序的EX-WSDL文件，生成Web服务行为模型。}

测试序列生成： 针对生成的行为模型使用不同覆盖准则生成测试序列，包括请求节点覆盖、响应节点覆盖及边覆盖准则，记录不同准则生成测试序列的个数、序列、覆盖元素。

测试用例生成： 针对上述生成的测试序列，采用状态覆盖与不采用状态覆盖两种覆盖准则生成测试用例，记录生成测试用例的个数。

测试执行与检测： {46%：利用MDGen工具执行测试用例并监控测试用例执行过程，错误调用服务的测试用例进行约束违规检测。} {49%：分析不同覆盖准则生成测试用例的约束违规检测能力。}

实验结果

PFC服务实验结果

WSDL扩展及部署：

{51%：扩展后的WSDL文档能够描述服务调用过程中存在的行为约束。} 根据5.2小节PFC服务描述可知login操作具有参数范围约束（表5-1约束）及重复调用约束（入库成功后不可重复调用入库操作）。 feeCalculate操作具有参数范围约束（表5-2约束）、参数关系约束（入库与出库操作的 车牌号一致、车辆出库时间大于等于入库时间）、顺序约束（入库成功后可调用出库操作） 及重复调用约束（出库计费成功后不可直接重复调用出库计费操作）。 扩展WSDL并部署，用以描述上述服务行为约束。

本文使用axis2开发部署PFC服务，部署成功后，访问URL： http: //localhost: 8080/axis2/services/ParkingFeeCalculator? wsdl获取EX-WSDL，如图5-1所示。

由图5-1可知，EX-WSDL文档能够正确描述服务存在的行为约束，在PFC服务中描述了参数范围、参数关系、顺序约束及重复调用约束。 并且，带有约束信息的EX-WSDL文档能够被正常部署与访问。

Web服务行为模型生成：

{42%：使用MDGen工具对上个步骤获取的EX-WSDL文档进行解析，生成的对应服务行为模型如图5-2所示。} 该模型符合PFC服务的行为约束，即基于EX-WSDL的服务行为模型生成技术能够正确的生成服务行为模型。

图5-1 PFC服务EX-WSDL文档访问

图5-2 PFC服务行为模型

测试序列生成：

{44%：根据不同的覆盖策略，得到满足一定覆盖准则下的测试序列；} 针对请求节点覆盖策略，PFC共产生4条测试序列，详细信息如表5-11所示； 针对响应节点覆盖策略，PFC共产生6条测试序列，详细信息如表5-12所示； 针对边覆盖策略， PFC共产生6条测试序列，详细信息如表5-13所示， Id表示序列编号、 Type表示序列类型、 Test Sequense表示序列内容、 Covered Elements表示序列覆盖元素。

表5-11请求节点覆盖策略测试序列（PFC服务）

表5-12响应节点覆盖策略测试序列（PFC服务）

表5-13边覆盖策略测试序列（PFC服务）

边覆盖策略测试序列（PFC服务）（续）

测试用例生成：

{45%：针对上述测试序列，不使用状态覆盖生成的测试用例个数与使用状态覆盖生成的测试用例个数如图5-3所示。}

图5-3 不同覆盖策略测试用例生成个数

横轴代表选取的覆盖策略，其中ReqN-S表示使用请求节点覆盖及状态覆盖准则； ReqN-NS表示使用请求节点覆盖但不使用状态覆盖准则； ResN-S表示使用响应节点覆盖及状态覆盖准则； ResN-NS表示使用响应节点覆盖但不使用状态覆盖准则； E-S表示使用边覆盖及状态覆盖准则； E-NS表示使用边覆盖但不使用状态覆盖准则。 纵轴代表生成测试用例个数。 {43%：Positive代表策略生成合规测试用例个数，Negative代表策略生成冲突测试用例个数。} 需要注意的是，使用状态覆盖准则生成的测试用例个数取决于服务操作的决策表中的规则个数。

测试用例执行与检测：

{48%：分别使用MDGen工具对六种覆盖策略生成的测试用例进行执行与约束违规检测。}其执行检测结果如表5-15所示。 其中ReqN-S、ReqN-NS、ResN-S、ResN-NS、E-S、E-NS分别代表使用的六种覆盖策略。 第一列代表测试用例违反的行为约束： paraRestriction、preOp、 Iteration及 paraRelation约束为服务 PFC执行过程中包含的行为约束（表5-10）， total表示使用 MDGen工具生成的违反服务行为约束的测试用例个数。根据表5-14可以看出：

六种覆盖策略均可发现全部服务调用错误并正确报告违反约束类型；

针对PFC服务，响应节点覆盖准则与边覆盖准则的行为约束检测能力高于请求节点覆盖准则，检测出了PFC服务中出现的所有行为约束。

{56%：表5-14 不同覆盖策略测试用例执行情况}

PFC2服务实验结果

根据5.2小节可知，为考虑由于需求的变化而进行的服务修改导致被调用服务接口处于停用状态的情况， 本文在 PFC服务中考虑时效约束生成了 PFC2，程序实现过程中删除feeCalculate操作功能，但并未对 PFC2的 WSDL接口进行修改， 用以模仿由于服务实现修改后带来的操作不可用问题。 使用axis2开发部署PFC2服务，部署成功后，访问URL：http: //localhost: 8080/axis2/services/ParkingFeeCalculator2? wsdl获取扩展后的服务WSDL。 由于删除 feeCalculate操作， PFC2服务不存在由于 feeCalculate操作带来的顺序约束、重复调用约束及参数关系约束， 即 PFC2服务仅存参数范围约束、重复调用约束（ login操作包含该约束）及时效约束。 使用axis2开发部署PFC服务，部署成功后，访问URL： http: //localhost: 8080/axis2/services/ParkingFeeCalculator2? wsdl获取扩展后的服务WSDL。 由于PFC2服务与PFC服务的扩展WSDL相同，因此生成的行为模型、测试序列与测试用例均与PFC服务实验结果相同。

测试用例执行与检测： {48%：分别使用MDGen工具对六种覆盖策略生成的测试用例进行执行与约束违规检测。} 其执行检测结果如表5-15所示。paraRestriction、Iteration及eTime约束为服务PFC2执行过程中包含的行为约束（表5-10）。根据表5-15可以看出：

六种覆盖策略均可发现全部服务调用错误并正确报告违反约束类型；

针对PFC2服务，响应节点覆盖准则与边覆盖准则的行为约束检测能力高于请求节点覆盖准则，检测出了PFC2服务中出现的所有行为约束。

{57%：表5-15 不同覆盖策略测试用例执行情况}

EXP服务实验结果

WSDL扩展及部署：

{51%：扩展后的WSDL文档能够描述服务调用过程中存在的行为约束。} 根据5.2小节EXP服务描述可知 calAmount操作具有参数范围约束（表5-4约束）及区域约束（该操作有固定局域网络访问权限， 实验机网络不在该操作权限范围内）。 airfareReimburse操作具有参数范围约束（表5-5约束）及区域约束。 totalAmount操作具有参数范围约束（表5-6约束）及调用约束（总金额计算操作调用车辆使用费计算操作及机票报销操作）。 扩展WSDL并部署，用以描述上述服务行为约束。

本文使用axis2开发部署EXP服务，部署成功后，访问URL： http: //localhost: 8080/axis2/services/ExpenseReimbursementSystem? wsdl获取扩展后的服务WSDL，该WSDL如图5-4所示。

图5-4 EXP服务扩展WSDL文档访问

由图5-4可知，扩展后的WSDL文档能够正确描述服务存在的行为约束，在EXP服务服务中描述了参数范围、区域约束及调用约束。 并且，带有约束信息的WSDL文档能够被正常部署与访问。

Web服务行为模型生成：

使用MDGen工具对上个步骤获取的扩展WSDL文档进行解析，生成的对应服务行为模型如图5-5所示。 该模型符合 EXP服务的行为约束，如 EXP服务所有操作的重复调用约束均为 true，即存在 e6， e11， e16边的执行，即基于扩展 WSDL的服务行为模型生成技术能够正确的生成服务行为模型。

图5-5 EXP服务行为模型

测试序列生成：

{44%：根据不同的覆盖策略，得到满足一定覆盖准则下的测试序列；} 针对请求节点覆盖策略，由于该服务不存在序列约束，因此未生成冲突测试序列。 EXP共产生3条测试序列，详细信息如表5-16所示； 针对响应节点覆盖策略，EXP共产生6条测试序列，详细信息如表5-17所示； {42%：针对边覆盖策略，EXP共产生条测试序列，详细信息如表5-18所示。}

表5-16 请求节点覆盖策略测试序列（EXP服务）

表5-17 响应节点覆盖策略测试序列（EXP服务）

表5-18 边覆盖策略测试序列（EXP服务）

测试用例生成：

{42%：针对上述测试序列，不使用状态覆盖生成的测试用例个数与使用状态覆盖生成的测试用例个数如表5-6所示。} 横轴代表选取的覆盖策略，其中ReqN-S表示使用请求节点覆盖及状态覆盖准则； ReqN-NS表示使用请求节点覆盖但不使用状态覆盖准则； ResN-S表示使用响应节点覆盖及状态覆盖准则； ResN-NS表示使用响应节点覆盖但不使用状态覆盖准则； E-S表示使用边覆盖及状态覆盖准则； E-NS表示使用边覆盖但不使用状态覆盖准则。纵轴代表生成测试用例个数。 Positive代表策略生成合规测试用例个数。 需要注意的是，

使用状态覆盖准则生成的测试用例个数取决于服务操作的决策表中的规则个数。

{55%：图5-6 不同覆盖策略测试用例执行情况}

测试用例执行与检测：

{48%：分别使用MDGen工具对六种覆盖策略生成的测试用例进行执行与约束违规检测。}其执行检测结果如表5-19所示。 paraRestriction、invokeOp及ipRegion约束为服务EXP执行过程中包含的行为约束（表5-10）。 根据表5-19可以看出：

六种覆盖策略均可发现全部服务调用错误并正确报告违反约束类型；

针对EXP服务，响应节点覆盖准则与边覆盖准则的行为约束检测能力高于请求节点覆盖准则，检测出了EXP服务中出现的所有行为约束。

{54%：表5- 19 不同覆盖策略测试用例执行情况}

EXP2服务实验结果

根据5.2小节可知，实验环境IP地址符合EXP2区域约束，即修改EXP的WSDL文档操作区域约束。 使用axis2开发部署PFC2服务，部署成功后，访问http: //localhost: 8080/axis2/services/ExpenseReimbursementSystem2? wsdl 获取扩展后的服务WSDL。由于修改了操作区域约束使得实验机IP满足操作区域约束要求，因此，EXP2服务仅存参数范围约束、及调用约束。 EXP2服务的扩展WSDL如图5-7所示。 与EXP服务相比，操作的区域约束不同。

图5-7 EXP2服务扩展WSDL文档访问

由于仅仅改变了区域约束内容，因此生成的行为模型、测试序列与测试用例均与EXP服务实验结果相同。

{48%：分别使用MDGen工具对六种覆盖策略生成的测试用例进行执行与约束违规检测。}其执行检测结果如表5-20所示。 paraRestriction、Iteration及eTime约束为服务PFC2执行过程中包含的行为约束（表5-10）。 根据表5-20可以看出：

六种覆盖策略均可发现全部服务调用错误并正确报告违反约束类型；

针对 EXP2服务，响应节点覆盖准则与边覆盖准则的行为约束检测能力高于请求节点覆盖准则， 使用状态覆盖比不使用状态覆盖准则的检测能力更高， ResN- S与 E-NS准则检测出了 EXP2服务中出现的所有行为约束。

{54%：表5-20 不同覆盖策略测试用例执行情况}

小结

本章使用两个 Web服务实例（每个实例两个版本）设计了实验，对于第三章提出的行为模型驱动的服务组合程序测试用例生成技术进一步进行了验证与评估。 根据实验结果可知： （1）EX-WSDL文档能够正确描述服务存在的行为约束，并且能够正常部署与获取；{41%：（2）基于EX-WSDL的服务行为模型生成技术能够正确生成服务行为模型；} （3）行为模型驱动的服务组合程序测试用例生成技术能够检测服务调用错误并正确报告违反约束；（4）响应节点覆盖准则与边覆盖准则的行为约束检测能力高于请求节点覆盖准则。

工作总结与展望

{42%：在面向服务的架构中，服务使用者只能依据服务规格说明（即WSDL文件）访问相关Web服务。} {48%：由于 WSDL文件中仅仅包含 Web服务接口的抽象描述，缺乏对服务提供操作语义信息的描述，} {43%： Web服务使用者难以了解 Web服务的正确使用方式，易于出现无法满足 Web服务的使用约束的情形，} 从而导致基于 Web服务的应用程序失效的问题。 本文通过在WSDL文件中引入Web服务行为相关的数据约束和控制约束描述，解决服务规格说明中缺乏服务行为逻辑描述而导致服务错误调用的问题。 提出了一种行为模型驱动的服务组合程序测试用例生成技术，开发了相应的支持工具。

分析与归纳了由于服务存在的隐含行为逻辑导致服务调用失效的情况，提出了时效约束、{43%：区域约束、序列约束、调用约束、参数范围约束、参数关系约束等6类服务行为约束类型；} 通过扩展标准的Web服务描述语言WSDL开发了EX-WSDL，支持上述6类约束的形式化表达。 并且，带有约束信息的WSDL文档能够被正常部署与访问。

{50%：提出了一种行为模型驱动的服务组合程序测试用例生成技术：} 通过解析基于EX-WSDL的服务规格说明，构建基于事件序列图的Web服务行为模型； 定义了请求节点、响应节点、边及状态4种覆盖准则； {67%：设计了满足不同覆盖准则的测试序列生成算法；} {48%：针对测试序列生成满足约束的测试数据，形成可执行的测试用例；} 本文方法生成的测试用例能够有效检测出违反上述服务行为约束的调用。

开发了行为模型驱动的服务组合程序测试用例生成工具MDGen，提高了行为模型驱动的Web服务组合测试用例生成技术的自动化程度。

采用2个Web服务程序实例验证并评估提出的技术的有效性与支持工具的实用性。

工作不足及未来展望：

设计并检验了所提出的行为模型驱动的服务组合程序测试用例生成技术，需要进一步与其他相关的模型驱动服务测试技术进行比较。

使用了2个Web服务进行实例验证，需引入其他的实例程序进一步评估技术的有效性。