关于序列约束（1）和参数范围约束（2）文献搜集：

* 基于扩展WSDL的测试用例自动生成\_袁雪莉 1 2
* Behavior modeling and automated verification of Web services 1
* 基于接口语义契约的Web 服务测试数据生成 1 2 白晓颖 软件学报
* Event- and Coverage-Based Testing of Web Services
* 用户需求驱动的Web服务测试 2
* 一种基于扩展WSDL的测试数据自动生成方法 2

调用约束

* Extending WSDL to Facilitate Web Service Testing (HASE 2002 )

时效，参数关系、区域

* 用户需求驱动的Web服务测试——有参数关系的
* 参数关系：一种基于合约式设计的测试技术研究

Constraints based Web Service Semantic Augmentation (ICWS 2014) 有问题 再读读

**同一个服务操作间的约束分类：**

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（可加入执行后状态约束、时间约束：一个操作执行完之后，多久内可执行其他服务）
* **时间约束**：一个消息的响应时间必须在多久内；
* **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：数值型【int，float, double, 】（最大值、最小值、枚举、默认值）布尔型（true/false、默认值）字符串（长度、最长、最短、枚举、默认）
* **组合约束**：某个参数选择后，另一个参数的选择就被确定了范围
* **计算约束**：一个操作的输出，参与到了另一个操作的计算中
* **输出约束：**

**不同服务提供的操作的约束分类：**

**错误：**

* **调用顺序错误（组合错误）（静态检测）**
* **调用参数不匹配（组合错误）：1.**参数赋值错误（a1=b1,a2=b2结果为a1=b2,a2=b1）**（静态检测）2.**使用人员输入的参数不符合该操作的约束要求、一个操作的输出不匹配另一个操作的输入约束**（动态检测）**
* **调用序列错误（组合错误）：**某个测试用例应该执路径1，但是现在执行了路径2
* **调用的服务不可用（服务错误）（动态检测）**
* **服务执行结果与约束不匹配（服务错误）**

**数据：**

* **有效的数据验证（约束求解）**
* **无效的数据（黑盒直接生成）**

**覆盖准则：**

**【基于扩展WSDL的测试用例自动生成\_袁雪莉】文献7 17**

方法：基于拓展的WSDL的测试用例自动生成（主要包括测试数据+操作序列两部分）

* Types元素添加了simpleType标签：用来描述数据类型和刻面约束
* Operation元素中添加了prep子元素，表示用来表示Operation的前置操作，分析扩展后文档中的操作依赖关系生成操作序列，结合测试数据，就可以有效地测试复杂的操作流程
* 简单测试数据生成：根据添加的数据类型，取值范围产生（随机值与极端值的考虑）
* 复杂测试数据生成：在简单数据类型中加入参数权重选择，根据组合生成
* 操作序列测试
* 四种覆盖准则
* Xml描述测试用例，及其重要性

**【WSDL-Based Automatic Test Case Generation for Web Services Testing】**

本文介绍了从WSDL自动生成Web服务测试用例的研究，测试用例从两个视角进行：单个操作的测试数据生成和测试操作序列

该框架针对一个原子服务的单个操作和其操作序列层面进行测试，并未考虑到组合服务层面

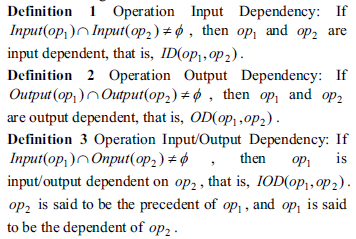
Test data are generated by analyzing the message data types according to standard XML schema syntax

Operation flows are generated based on the operation dependency analysis（由操作约束生成操作序列）The sequence specifications can be used for path testing.

三种类型依赖关系定义：

* 输入依赖性ID Two operations are said to be input dependent if they both share a set of messages as their input message.
* 输出依赖性IOD two operations are said to be output dependent if they both share a set of messages as their output message.
* 输入/输出依赖性OD An operation op1 is said to be input/output dependent on anther operation op2 if at least one of the input messages of op1 is the output message of op2.

We use *Input (op)* to represent the set of input messages of an operation *op* and *Output (op)* as the set of output messages, the definitions are given as follows:



最后，该生成的测试用例被记录在基于XML的测试文件中

文章认为test tools need to analyze the specifications to extract necessary information such as interface operations（操作接口）, data structures（数据结构）, and operation semantics（操作语义）.

WS测试工具也开始出现在最近几年[ 14 ] [ 15 ] [ 16 ] [ 17 ] 这些工具

便于监测和WS执行调试，如跟踪过程中，调试WSDL和SOAP消息，捕获和回放的请求和

响应，并检查语法和语义规格的正确性。

步骤：

* The specification of the service is first parsed into a DOM tree representation based on the WSDL schema.
* Test cases are generated from four levels: **test data generation（测试数据，针对每个type的）**, **individual test operation generation（针对单个操作的）**, **operation flow generation（操作流程的）**, and **test specification generation（测试脚本XML规范）**.
* 测试数据根据type数据分析包括简单的数据类型，复杂的数据类型、聚合数据类型和用户派生数据型
* 个别操作的测试由分析相关的参数（即消息）从测试数据中组合
* 一个服务可能包含多个操作。测试组合函数，生成测试操作序列。生成序列基于操作依赖分析，包括输入依赖，输出依赖和输入/输出依赖

该框架由几个部分组成：

* 测试用例生成器：接收提交的WSDL生成测试用例，储存在中央数据库
* 测试执行控制器：控制测试执行，从测试数据库检索测试用例、监视测试运行和搜集测试结果
* 测试分析器：分析测试结果，评估质量（那么我们可以看看组合的程序违背规则情况占总测试条目的百分比，给一个初步的质量评估），产生测试报告
* 针对服务提供商、服务代理、服务消费者对于数据库的访问有不同权限



**操作序列生成：**

**先对每个操作分析其输入输出约束，得到其前置操作（PRE）及其后置操作（DEP）集合，**

**【Extending WSDL to Facilitate Web Services Testing】**

**W.T.Tsai针对This is mainly caused by the insufficient information provided by the WSDL file of a web service 问题 .提出了四方面的扩展机制（依赖信息的缺乏）**：从WSDL文件中，我们不能得到对依赖性等测试有用的信息

* 对输入输出依赖性的描述input-output dependency（web服务之间的输入输出依赖）描述了接口之间的关联
* 对服务间调用序列的描述invocation sequence（Method Sequence Specification）WSInvocationDependenceType to represent the callers and callees

捕获服务间的数据流和控制流

WSInvocationDependenceType to represent the callers and callees. There are two sub-elements in it, WSICallers and WSICallees of WSICallersType and WSICalleesType respectively. For both the WSICallersType and WSICalleesType, there is a link to their WSDL files for reference. By tracing this information among participating web services, it is possible to generate the complete calling sequence, and the calling sequence is useful in path testing and data flow testing.

* 对并发序列的描述concurrent sequence specifications（服务或操作是并发的（都不做或者选择多个））
* 对功能层次关系的描述hierarchical functional description

Testing web services is equivalent to black-box testing where specifications are available but no design or code are available, and the specification is written in WSDL, while code can be written in Java, C#, or other programming languages.

目前的WSDL文档仅仅包含如下信息：

* 输入输出参数个数
* 输入输出参数类型
* 输入参数排序
* 服务调用地址

**【基于扩展 WSDL 行为描述的Web 服务组合】**

**【Behavior modeling and automated verification of Web services】**

特别是，我们建议将Web服务行为分为两种类型：操作行为（操作之间定义的约束）和控制行为（执行到了哪一步）

The operational behavior shows the business logic that underpins the functioning of a Web service

In contrast, the control behavior guides the execution progress of the business logic of a Web service. The control behavior relies on a number of states (activated, not-activated, done, aborted, suspended, and compensated) that are reported in the transactional Web services literature [22,23,38] 激活的，未被激活的，完成的，中止的和补偿的

使用**符号模型检验**进行web服务验证symbolic model checking‘

**【Adequate Monitoring of Service Compositions】**

在面向服务的应用程序中需要有手段来评估被监测服务的互动性

**【WSCMon: runtime monitoring of web service orchestration based on refinement checking】**

提出了监视BPEL服务组合程序的方法，在WSCMon，系统规格提供了一个抽象的模型，所有的复合Web服务的所有可能的行为。

使用CSP（Communicating Sequential Processes）描述流程One ofthe transformations is intended to generate CSP processes as system specifications from BPEL processes

使用LTL（线性时序逻辑）表达需求规格

检查BPEL流程执行与规格之间的不同

An attractive complement approach of testing and verification is runtime monitoring，In this approach, composite services are monitored at runtime while execution traces are generated. Then, the conformance of generated traces is verified against specifications that show intended behaviors

**【组合服务交互测试研究综述】**

用于组合的Web服务是已经存在的基本服务，被称为原子服务，它们对组合服务的用户是透

明的；组合而成的新Web 服务被称为组合服务，是对其他服务的合成，并以接口形式提供给用户或者其他服务使用。

组合服务按照组合服务流程，分别与各个原子服务进行直接交互，通过对原子服务的访问，实现对细粒度功能的聚合。组合服务与原子服务的交互正确与否，决定组合服务能否将细粒度的功能正确聚合，从而获得粗粒度的功能。因此，组合服务的交互测试是组合服务测试的一个重

要方面。

以BPEL4WS 作为组合服务实现方式，从BPEL4WS的执行引擎中获得运行时数据[27-32]；

**【基于接口语义契约的Web 服务测试数据生成】**

模型包含被测服务的输入/输出参数、控制依赖、数据依赖等信息,有关输入/输出参数的数据本体模型包含了被测服务丰富的领域信息.在数据本体模型的基础上,利用约束规则定义多数据间相互依赖和限制关系,建立约束模型.

文献[13]将UML2 中的协议状态机描述引入到Web 服务描述WSDL 中,增加对服务前置/后置条件以及数据/进程约束的描述,增强了Web 服务的可验证性

XML 级契约对WSDL 等基于XML 的Web 服务协议进行了扩展,增加了服务操作方面的信息. 模型级契约提供了服务契约的可视化形式,方便了开发人员之间的交流与合作.该研究将服务的行为信息加入到Web 服务描述中,实现基于契约的Web 服务测试,从服务提供商和用户角度分别定义了提供契约和请求契约,描述了提供商提供的服务行为和用户需要的服务行为.

双方对契约理解的一致性是服务能够正确使用的基础.然而,服务提供时通常有适合的应用范围和场景,例如参数的取值范围、操作的调用顺序等;服务使用也建立在对服务的某些隐含假设之上.现有的服务接口描述中缺少足够的信息,无法准确表达这些隐含的约束限制条件,常常造成双方对服务理解的不一致,从而导致运行错误

建立接口的契约模型.采用基于谓词逻辑表达的规则语言,描述服务的数据及流程的约束限制条件.契约模型定义了原子服务的功能以及输入/输出参数,规定了服务之间的数据依赖关系和控制依赖关系,为服务调用以及服务之间的协同合作提供了所需的场景信息,并定义了在相应场景下的预期结果;

**【Automatic WSDL-guided Test Case Generation for PropEr Testing of Web Services】**

我们的工作的目的是在全自动测试的Web服务，理想情况下，用户指定web服务预期满足的属性（以输入输出关系的形式），

**一种基于属性的测试（PBT）**工具

随机测试用例的操作并检查其响应。

我们的工具可以选择允许用户可以很容易地修改它的输出添加语义信息生成器或写属性，测试更多涉及的Web服务的功能

该工具的框架：

从URL获取WSDL文档，该文档被传输到xmerl工具：分析与WSDL规范的相关信息；Yaws工具：提取所有与操作有关的信息；这两种信息用来合成最后的测试用例

之后用户可以修改自动生成的测试文件，添加自己的属性或细化该测试文件

最后该文件作为输入，在PropEr工具，调用Web服务并对结果进行了分析

**基于属性的测试：**用户指定的有效输入的一般结构，以及与预期保持有关的输入-输出关系的属性。PropEr [10] is an Erlang-based PBT tool 它可以接收该关系属性信息，并逐步创建复杂的测试用例，执行它们，然后监视响应，以确保它符合指定的属性。此外，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

属性是用宿主语言编写的

为Web服务测试的正确使用，首先我们需要指定将用于调用Web服务操作的SOAP消息的结构，然后，我们需要一个属性，将收到Web服务响应

Revising WSDL Documents:Why and How, Part 2

**【A pattern for WSDL-Based Testing of Web Service Compositions】**

It is realized by various software tools like SoapSonar (Esposito 2006), SoapUI (SmartBear), Parasoft SOAtest (Parasoft Corporation), etc.

本文提出了一种模式的基于WSDL的Web服务测试。集成在了TASSA（Pavlov et al. 2010）一种用于复合Web服务的测试的平台，该工具提供了基于Web服务的WSDL功能测试的测试用例的生成和执行。

*  Missing of graphical user interface of business process;
*  Invocation of services which are external for business process under test;
*  Need of additional efforts and tools for testing third party/external web services of the business process in order to validate their quality;
*  Possibility for usage of particular web service by multiple business processes;
*  Security issues established in distributed environment such as authentication, authorization, data integrity and privacy, etc.

提到了目前测试的状况：In addition, the presented tools support only black-box testing, giving information about the final result from the execution of the business process. If a particular test case fails, the testers are able to analyze only the fault data in the SOAP response having no idea what actually happens with the data flow of the orchestrated web services.**（测试用例失败后的定位问题）**

解决方法从三个角度出发考虑：

Test case design (1); Test case generation (2); Test case execution (3).

XML Schema基本架构

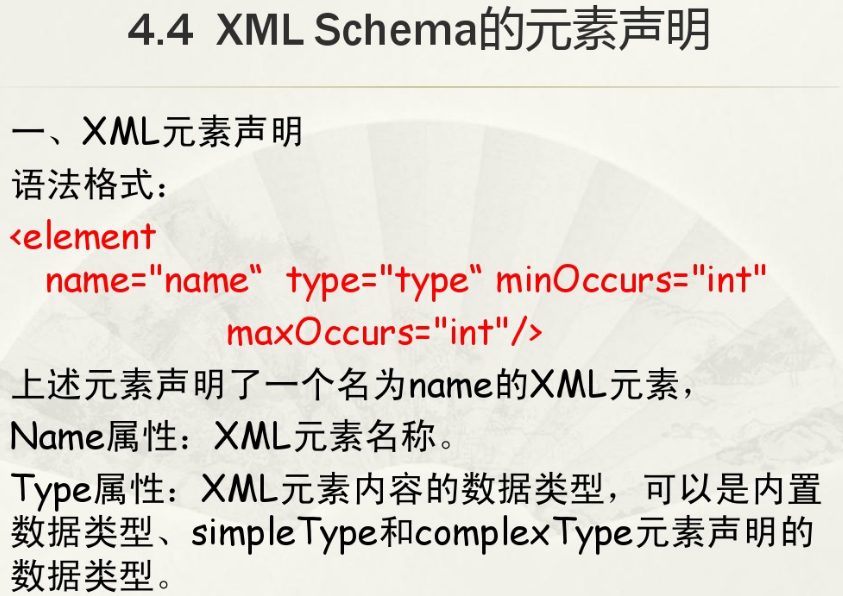
|  |  |
| --- | --- |
| 元素 | 说明 |
| Schema | Xml Schema的根元素 |
| Element | 声明xml元素 |
| SimpleType | 用户定义的数据元素   * 本身类型是默认类型但加入了范围约束 * 本身类型是用户自己定义的类型，该参数由其他默认类型的参数集合形成 |
| ComplexType | 声明XML元素的子元素 |
| attribute | 内置数据类型，声明元素属性（WSDL里面没有这种内置） |
| element | 内置数据类型，声明元素属性  <element name="addReturn" type="xsd:double"/> |

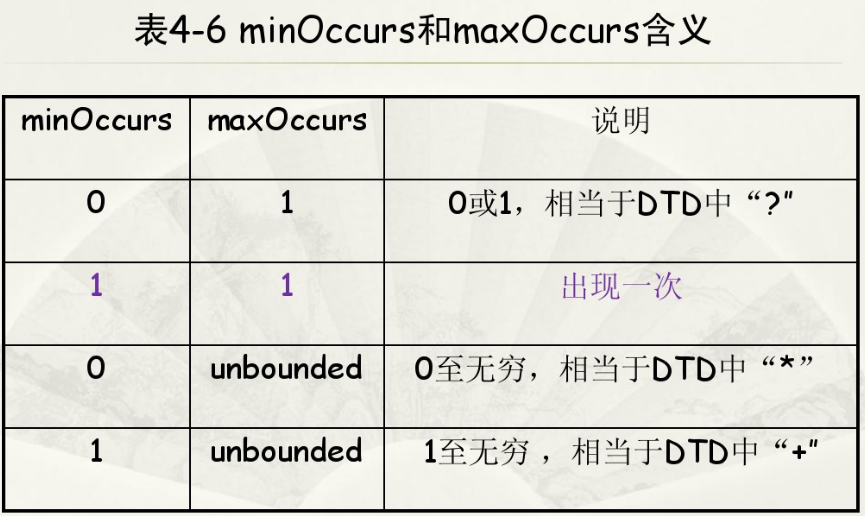












**[Automatic Generation of Test Data for XML Schema-based Testing of Web Services] 从xml中自动生成测试数据**

**【Constraints based Web Service Semantic Augmentation】**

服务约束 对象约束

service relations are identified on the basis of IOPE (input, output, precondition and effect).

【Constraint-Aware Approach toWeb Service Composition】

Web服务组合方法约束意识

约束包括：Service constraints

服务内涵和服务扩展：

服务有效时间/可用区域/服务针对不用用户级别的特殊性

通过一些现实世界的商业场景，我们描述的问题。然后，定义了服务约束和形式表达式。

可以再看看该篇文章的第49参考文献：每个web服务都与一个或多个语义关联本体中的概念，每个概念可能有多个属性.因此，服务提供商可以施加约束一些属性通过限制其范围，以便使服务功能明确，确保其正确执行[ 49 ]。这个属性值的范围可以通过数值表示值或一组语义概念。形式化表达服务限制如下。

再看下该篇文章的参考文献40

使用三个元素来表示不同限制： Attribute(代表服务属性) Opr（操作符号，如= ≠ ＞，not in等） InstanceData

可以看看有关SAWSDL与SWRL的相关文章【40】与【46】

\*\*\*\*\*\*【Extending WSDL to Facilitate Web Services Testing】

**W.T.Tsai针对This is mainly caused by the insufficient information provided by the WSDL file of a web service 问题 .提出了四方面的扩展机制（依赖信息的缺乏）**：

* 对输入输出依赖性的描述input-output dependency（web服务之间的输入输出依赖）描述了接口之间的关联
* 对服务间调用序列的描述invocation sequence（Method Sequence Specification）WSInvocationDependenceType to represent the callers and callees （客户端调用WS1，WS1又调用了WS2）

捕获服务间的数据流和控制流

WSInvocationDependenceType to represent the callers and callees. There are two sub-elements in it, WSICallers and WSICallees of WSICallersType and WSICalleesType respectively. For both the WSICallersType and WSICalleesType, there is a link to their WSDL files for reference. By tracing this information among participating web services, it is possible to generate the complete calling sequence, and the calling sequence is useful in path testing and data flow testing.

* 对并发序列的描述concurrent sequence specifications（服务或操作是并发的（都不做或者选择多个））（该文章并未提及）
* 对功能层次关系的描述hierarchical functional description
* 序列描述（许多WSDL中存在的序列）

Testing web services is equivalent to black-box testing where specifications are available but no design or code are available, and the specification is written in WSDL, while code can be written in Java, C#, or other programming languages.

目前的WSDL文档仅仅包含如下信息：

* 输入输出参数个数
* 输入输出参数类型
* 输入参数排序
* 服务调用地址

\*\*\*\*\*【Web Service with Criteria: Extending WSDL】

该文章认为WSDL描述了接口的功能与非功能属性，该文章关注从非功能属性扩展WSDL

Web服务有三个特点：功能，行为与非功能[其中第五篇参考文献]

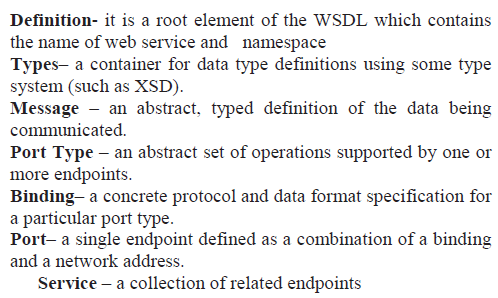
**扩展了WSDL中的definition和service标签**

更加详细的描述了这个服务提供的服务可以进行依据什么因素进行的执行。**更好的进行服务的搜索：**例如：一个宾馆预订服务，在服务中添加以“包厢”和“有网络”两个标准。

近年来，一些研究人员延长了WSDL指定属性不能用WSDL标准。许多努力已作出延长WSDL。这些范围从指定的安全要求，QoS，性能，测试参数处理版本。（In recent times, several researchers have extended the WSDL to specify properties which cannot be expressed using the standard WSDL. Many efforts have been made to extend WSDL. These range from specifying security requirements, QoS, performance, testing parameters to handling versioning.）

看看该文章的参考文献14、15、16 (ok)、17

WSDL解释：WSDL forms the basis for Web Services. According to W3C, WSDL which is in XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information [25].



[A flexible extension of WSDL to describe nonfunctional attributes]

Web服务描述语言（WSDL）不能描述一个对象它的所有信息，如WSDL没有提供方法描述服务的非功能方面，而只专注于功能部件。本文提出了一种灵活的扩展WSDL描述非功能属性。在这种方式中，WSDL可以以服务的形式完全描述一个物理对象。

However, WSDL cannot finish this assignment since it provides no method for representing non-functional aspect of a service while it only focuses on functional parts.

[WSDL and BPEL extensions for Event Driven Architecture]

**【一种Web 服务的测试数据自动生成方法】**

文章基于合约式设计的Web服务测试技术，提出一种Web服务的测试数据自动生成方法. 首先根据WSDL文档采用随机法自动生成初始测试数据，然后使用合约变异技术进行测试数据的选择，据此可以生成一组达到一定合约变异充分度的有效测试数据，从而提高Web 服务的测试质量和效率。最后实现了一个Web 服务的测试数据自动生成工具原型，并通过实验验证了方法的有效性。（该文章中参考文献3、8）

基于合约式设计的Web服务测试技术：

通常，服务使用者在使用Web服务的过程中会遇到如下的问题:一是无法完全了解Web 服务的正确使用方式，从而在测试时不能对发现的错误进行准确的定位，即服务使用者无法分辨究竟是Web服务本身的错误，还是使用Web服务的方法有误；二是很难测试Web服务是否符合使用需求。

合约表明了服务提供者和服务使用者相互的义务和利益，可用于区分软件失效时的责任：如果前置条件被违反，则应该在使用者处寻找错误；如果后置条件被违反, 责任在提供者

根据前置条件（操作参数的数值约束）生成初始测试用例集：例如, 有某int类型的参数i，其Cpre(i>=0)∧(i<=100), 用户定义在有效等价类中生成6个测试数据, 在无效等价类中生成2个测试数据，执行算法后，i产生的测试数据是{- 1，-587，0，1，100，99，41，7，101，513}。规约变异的目的不是为了发现规约中的错误，而是为了发现由于规约被错误理解或实现所导致的程序中的错误.因此, 对服务提供者的合约变异可视为提供者对设计规约的错误理解, 对服务使用者的合约变异则体现了使用者的不同需求. 我们通过对Web 服务的合约进行变异, 从中选择测试数据.

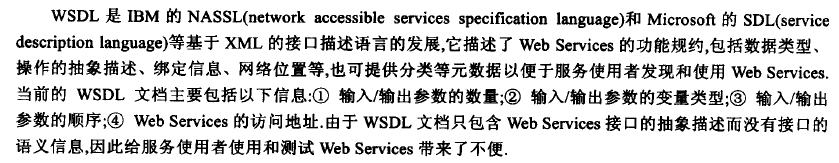
输入参数个数过多导致组合后的测试用例数量过大（笛卡尔积）——因此要进行精简：减少测试数据范围但是保持其数据的有效性。

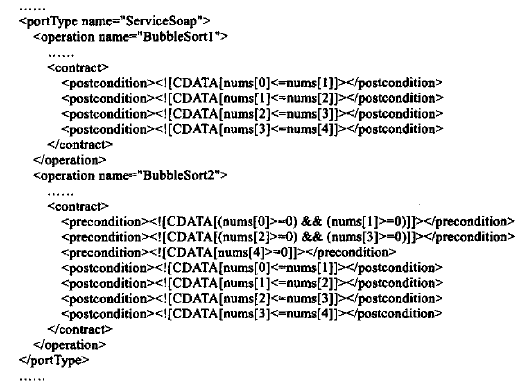
Web 服务的变异系统为WS \_MAS =〈WS ,CM , MO〉，其中，WS代表Web服务；CM代表Web服务合约的变异算子；MO 则是变异预言(Mutat ion Oracle)。变异预言是区分原程序与变异体的程序或条件[11]。

其中的参考文献13需要看一看

**【一种基于合约式设计的测试技术研究】**

服务提供者指定的合约：前置条件描述了使用Web Service必须满足的条件，服务提供者的后置条件相当于对Web Service的部分功能描述

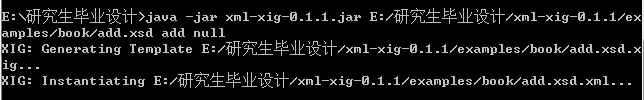




【XIG从XSD文件中自动上生成XML文件】

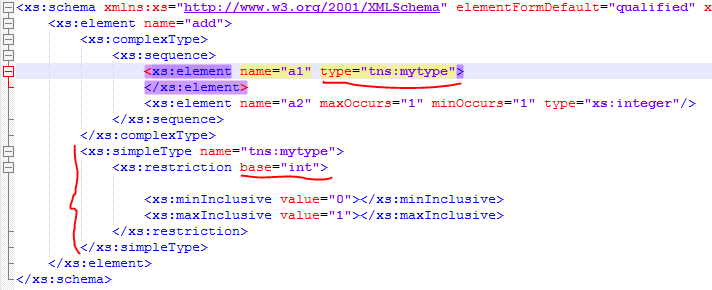
E:\研究生毕业设计>java -jar xml-xig-0.1.1.jar E:/研究生毕业设计/xml-xig-0.1.1/examples/book/add.xsd add null

在xig的jar目录下进行java -jar xml-xig-0.1.1.jar xsd文件 节点名 null 的执行

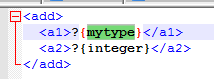


注意

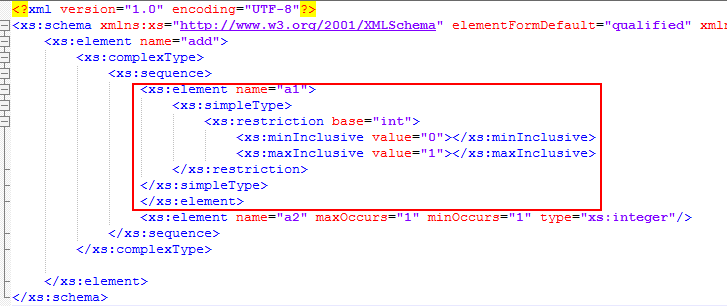
* 用XIG分析的文档有一个问题：标签需要添加“xsd:”前缀！！！
* 不能分析出attributes、choice标签
* 当simpleType在元素外，不能分析到它的基本类型



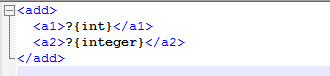
结果：



如果：



结果：



【调用SOAPUI测试web服务】

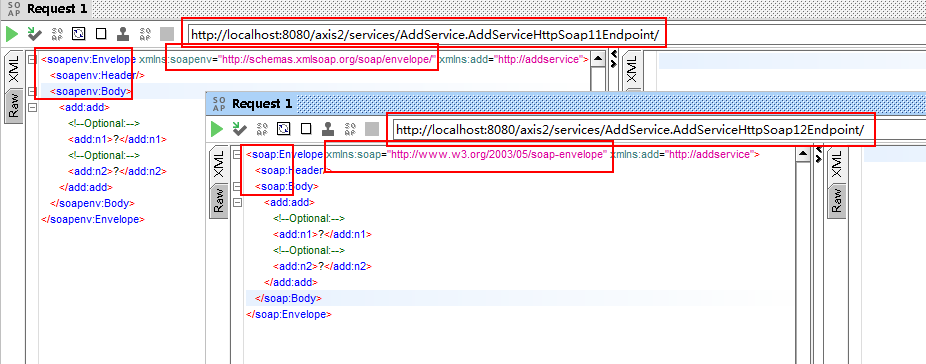
**调用代码请看自己的github上的Proj1\Significant\** **SoapInvoker.java和InvokeTest.java**

WSDL SOAP1.1与SOAP1.2：用soapUI分析WSDL得到的测试框架中：

Soap1.1接口的标签为soapenv，地址为：<http://schemas.xmlsoap.org/soap/envelope/>

Soap1.2接口的标签为soap，地址为：

http://www.w3.org/2003/05/soap-envelope



【Axis2及其生成web服务（WSDL发布及使用URL访问方法）】

介绍：

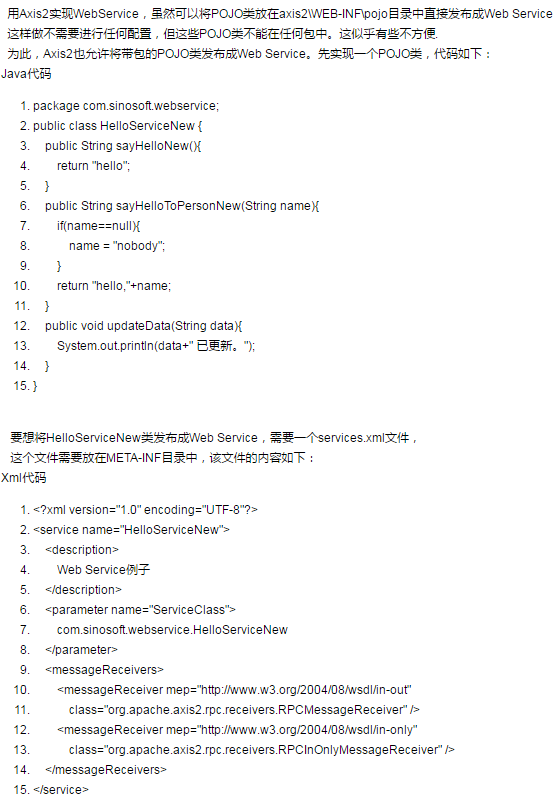
Axis2是下一代 Apache Axis，具有更强的灵活性并可扩展到新的体系结构。Axis2 基于新的体系结构进行了全新编写。

特点：

* 利用新的XML解析器提供的灵活性按需构造对象模型
* 支持HTTP，SMTP，JMS，TCP传输协议。
* 支持REST (Representational State Transfer)。
* 支持不同的消息交换模式。目前Axis2支持三种模式：In-Only、Robust-In和In-Out。In-Only消息交换模式只有SOAP请求，而不需要应答；Robust-In消息交换模式发送SOAP请求，只有在出错的情况下才返回应答；In-Out消息交换模式总是存在SOAP请求和应答。

使用AXIS2生成web服务（aar）及其WSDL【使用services.xml配置文件发布】

请看网站：<http://blog.csdn.net/xiaoxiaoxuewen/article/details/7054056>





**我用的方法：**

**先在eclipse：里面建立Java project，之后写Java文件，之后导出jar，向压缩文件中的META-INF文件夹中添加手动services.xml，修改jar后缀为aar，将其放置到tomcat6\webapps\axis2\WEB-INF\services里面即可**

使用AXIS2调用web服务;

导入如下包：

**import** org.apache.axis2.AxisFault;

**import** org.apache.axis2.addressing.EndpointReference;

**import** org.apache.axis2.client.Options;

**import** org.apache.axis2.client.ServiceClient;

其他的请看自己的github上的Proj1\Significant\SoapSend.java

使用URL调用web服务：

Web url/operation?参数1=value&参数2=value&参数3=value……

# 2017年3月30日星期四讨论问题

* 考虑输入参数组合约束（王可师兄 与或非 因果图）
* 假设前提是：提供的约束都是正确的，如果调用违反约束，直接拦截
* 工作意义：（1）提供了一种手段：拦截错误的传播，即如果不加入约束，实际上这个错误是被传播下去了，导致最后程序的崩溃或者执行错误。（2）基于上述提供的手段进行测试，如果违反了，我可以测出来：约束定义全一些，针对每种约束都有应对手段，都可以测出来即可
* 通过案例说可以做出来~各种各样的情况~技术工具可以处理检测出来即可

选题目的及意义：

借用一个标准把我们理解的有可能的这些约束标准化的表达出来，依据其做测试 把缺失的信息用一个桥梁表达出来，用该手段可以做进一步的测试。（总体思路是用约束表达）

**从服务组合层面对web服务的正确使用进行考虑**

如何从缺失信息与模型测试连接：**考虑操作有依赖——引入模型测试**

**针对非本地wsdl的测试：可能要把约束信息定义在document标签里面，先去看看@机制以及为什么目前网络访问wsdl显示不出来simpleType约束，如何显示出document标签**

WSDL可拓展机制：wsdl document标签 可以在里面定义约束，这样远程调用wsdl也可以获取到

我们通过接口的可扩展元素提供URI引用机制



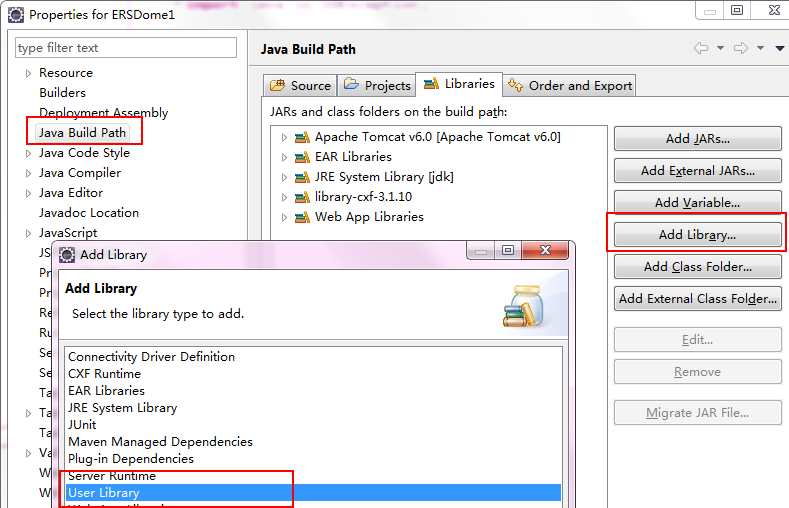
# 2017年3月31日星期五(cxf生成web服务)

向工程中加入User Library：

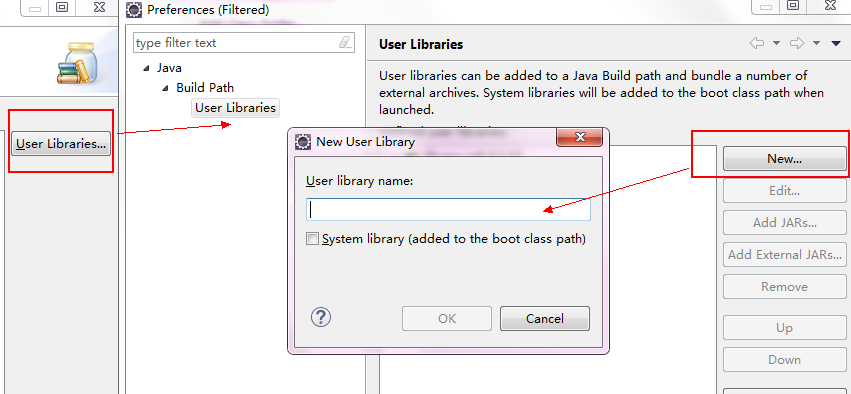
* **在目标工程中，右键，Build Path -> Configure Build Path 中的Java Build Path –> Add Library**



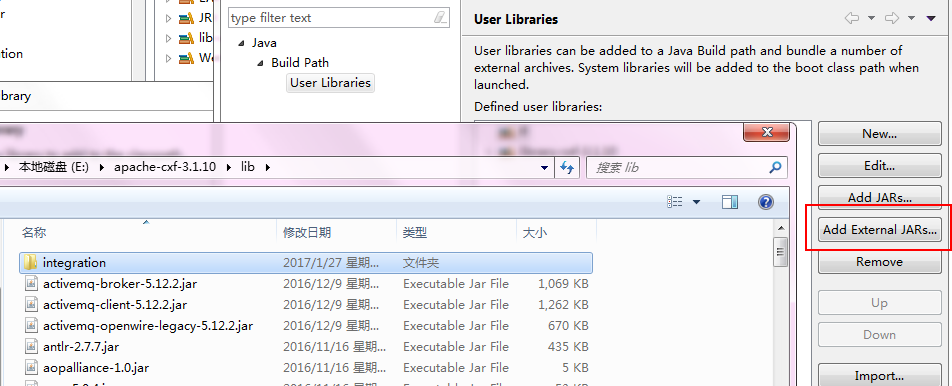
* **选择User Library**



* **点击new生成新的User Library并命名：我们命名library-cxf，点击ok**



* **之后选中新生成的空User Library，点击Add External Jars，选择要加入的jar，点击打开即可，之后一路就点击ok finish这种按钮即可**



1. CXF方式的WebService发布有四种:
2. 1.使用CXF内置的Jetty发布WebService
4. 2.使用RI(Oracle JAX\_WS RI) 发布WebService
6. 3.在Web容器(Tomcat)中基于Spring发布WebService
8. 4.CXF不基于Spring直接发布到 Tomcat

# 2017年4月2日星期日（使用Axis2及自定义wsdl生成、发布web服务）

**使用Axis2生成web服务（我们最后认为十分不错的选择~~）**

* **准备工作：**

**环境配置为tomcat6 以及axis2 1.6.2**

**在eclipse中安装axis2插件，使其支持axis2的web服务的发布及从WSDL文档生成services.xml文档的功能**

**由于我们使用的axis2的版本为1.6.2 因此我们下载的安装插件的版本也应该为1.6.2 防止由于版本不对应的问题引入的错误**

**第一步：下载axis2-1.6的插件压缩包（下载的文件我放到了“E:\eclipse安装的axis2插件”这个目录下），axis2-eclipse-codegen-plugin-1.6.2.zip 和axis2-eclipse-service-plugin-1.6.2.zip；**

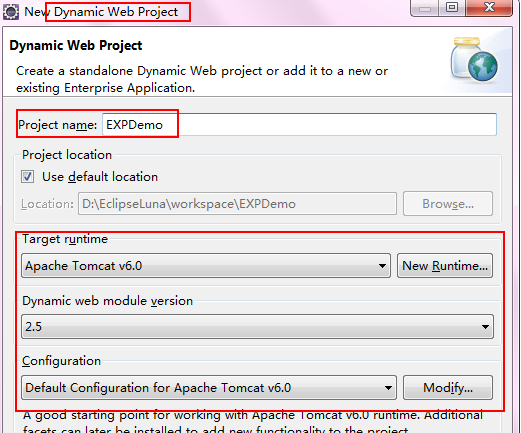
**第二步：解压下载的两个压缩包，并且将解压后的jar放到eclipse的dropins目录下即可；**

**第四步：重启eclipse，在新建工程时，选择new-->other，然后在弹出窗口中输入axis2，如果出现axis向导，则代表安装成功.**

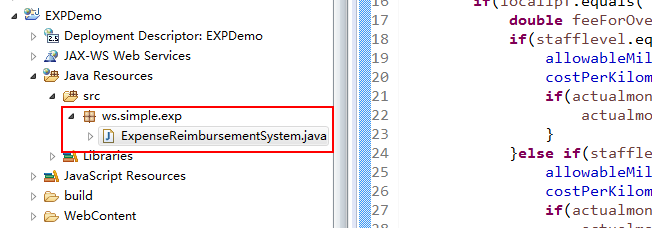
**之后可以进行服务的发布（aar包的生成和wsdl生成services.xml）**

* **生成aar包：**

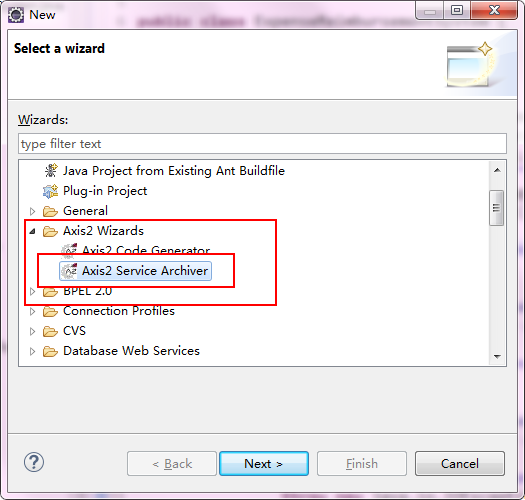
**在eclipse中new一个Dynamic Web Project，取名EXPDemo。编写一个简单的EXPDemo服务器代码**



**进行服务业务逻辑代码的编写：**



**在eclipse的空白workspace处，右键new->Other,在弹出的对话框中，找到Axis2 Service Archiver,双击->选择EXPDemo所在项目的class路径，如下图所示：**



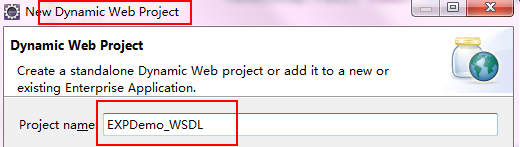


**点击next，添加WSDL文档：这里的文档是从正常的eclipse中的动态web项目使用jax-ws自动生成，人工修改加入约束的WSDL！！！**

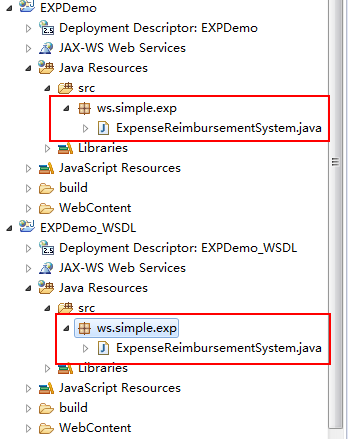
**其中，十分重要的是进行如下修改：**

1. **添加**<wsdl:documentation></wsdl:documentation>**以及其中的约束信息，用以表达服务正确执行的额外约束**
2. **添加针对操作的输入参数的约束，最大最小值，枚举值，正则表达式等等**
3. **修改自动生成的wsdl的wsdl:service的名字，改为对应的服务实现文档的类名即可！！！**
4. **修改wsdlsoap:address标签的location属性！！修改为**[**http://localhost:8080/axis2/services/portname**](http://localhost:8080/axis2/services/portname) **portname为service的port端口的名字**
5. **下面介绍下从正常的eclipse中的动态web项目使用jax-ws自动生成初始wsdl：**

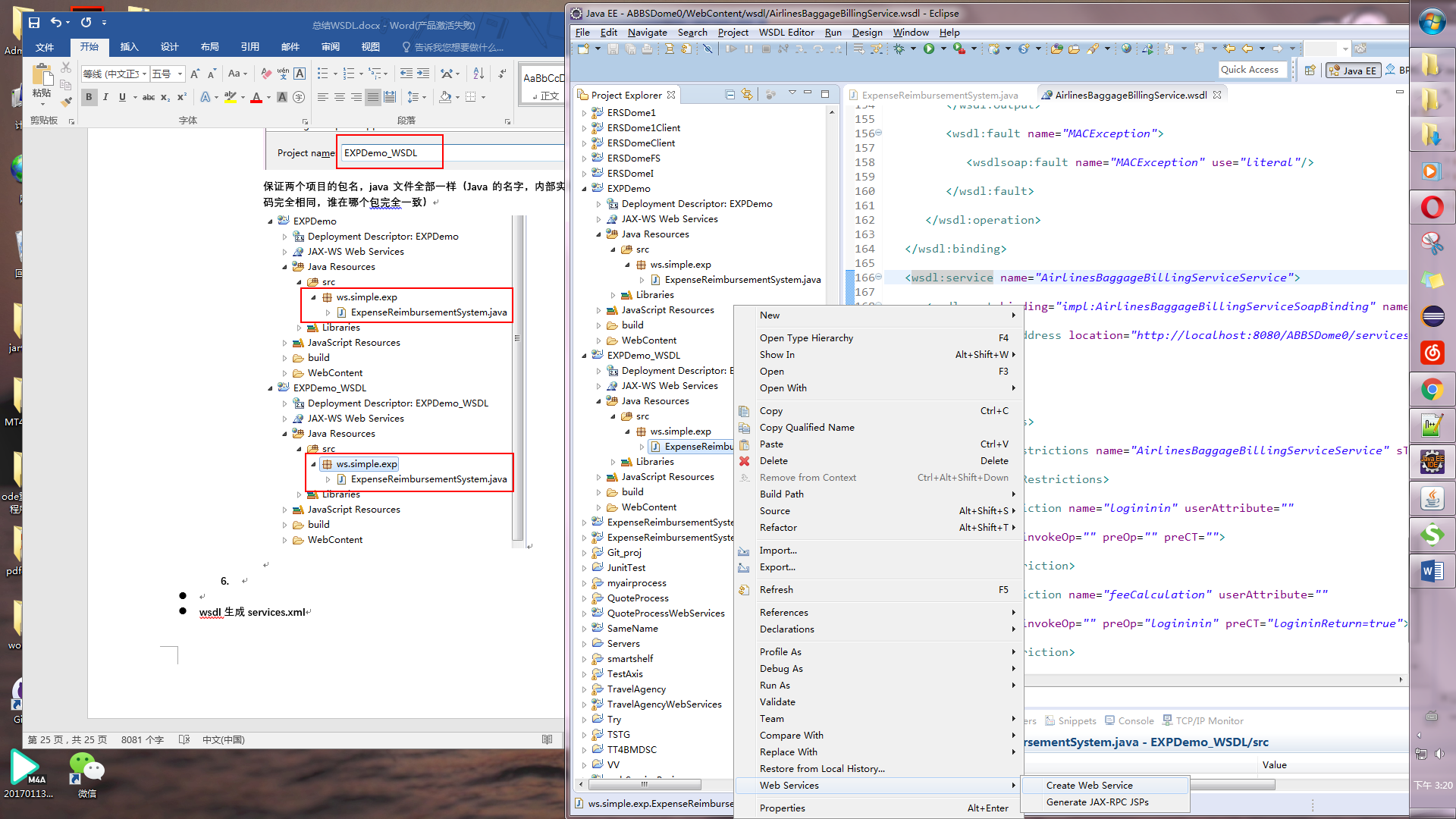
**先建立一个相关的有wsdl命名的动态服务，配置和之前项目的一样！！**



**保证两个项目的包名，java文件全部一样（Java的名字，内部实现的代码完全相同，谁在哪个包完全一致）**



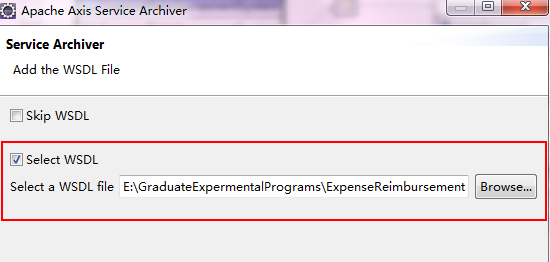
**针对该项目的java文档，右键使用eclipse自带的create web service 生成服务而不部署，这样可以生成初始的wsdl**



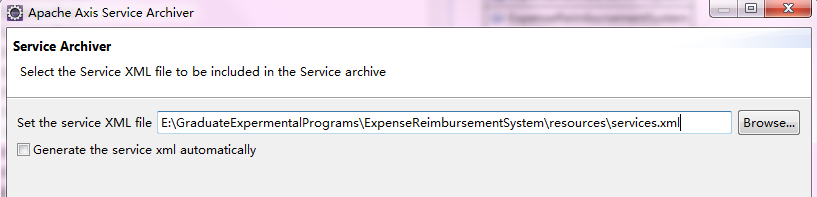
**针对初始的WSDL进行约束的添加即可**

1. **修改完wsdl后留用**

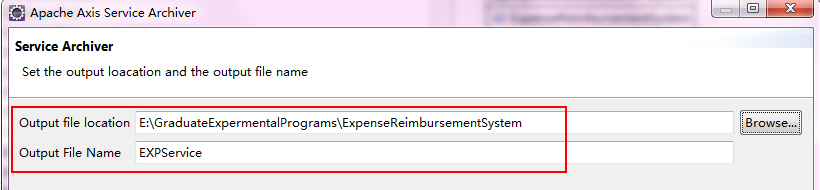
**点击next后，选择添加本地wsdl文档！！**



**之后一路next~知道选择service.xml，点击有WSDL生成的services.xml（这个文档如何生成之后会解释）**



**选择导出aar的存放位置，finish即可**



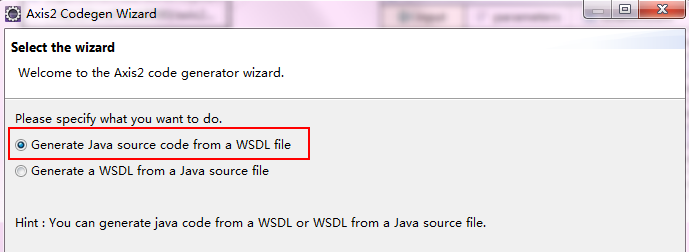
**将aar放在Tomcat 6.0\webapps\axis2\WEB-INF\services下面，启动服务器，输入**[**http://localhost:8080/axis2/services/ExpenseReimbursementSystem?wsdl**](http://localhost:8080/axis2/services/ExpenseReimbursementSystem?wsdl)**即可访问wsdl**

* **wsdl生成services.xml**

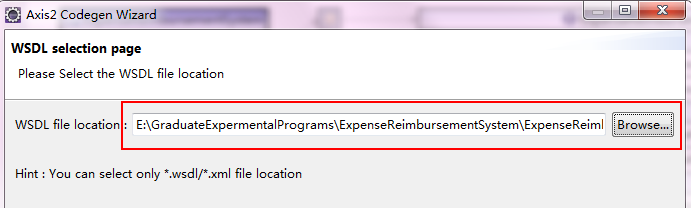
**打开eclipse点击file new 选择Axis Code Generator**



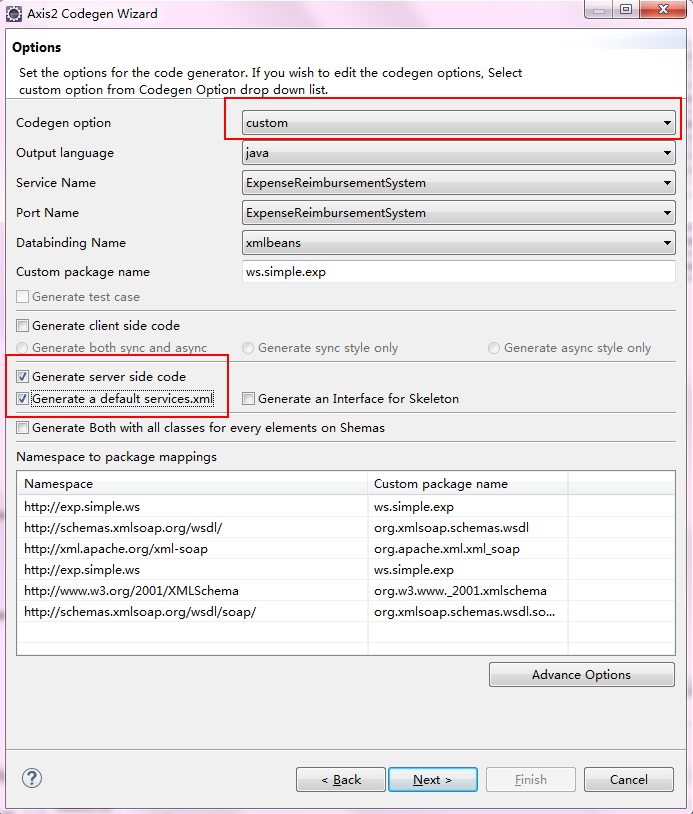
**选择从WSDL生成java code**



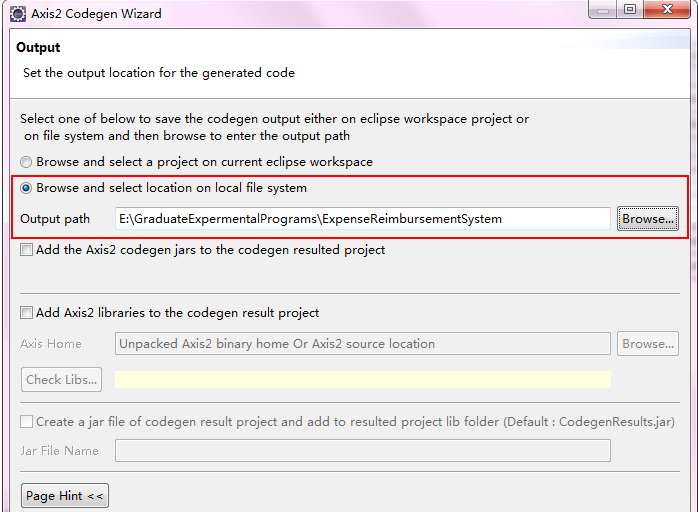
**选择项目存放位置**

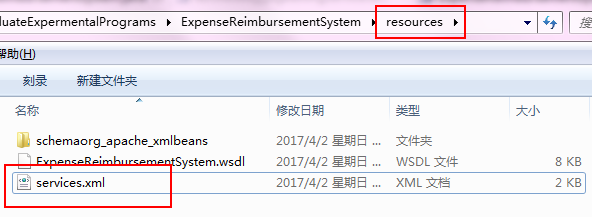


**选择custom模式，选择生成服务端，生成默认的services.xml !!!!非常重要呀**



**点击next选择存放位置**





**可以看到已经生成了初始的services.xml对其进行两个位置的修改：将**

|  |
| --- |
| **<messageReceivers>**  **<messageReceiver mep="http://www.w3.org/ns/wsdl/in-out" class="ws.simple.exp.ExpenseReimbursementSystemMessageReceiverInOut"/>**  **</messageReceivers>** |

**中的class修改为org.apache.axis2.rpc.receivers.RPCMessageReceiver**

**将**

|  |
| --- |
| **<parameter name="ServiceClass">ws.simple.exp.ExpenseReimbursementSystemSkeleton</parameter>** |

**修改为Java程序实现过程的class文件路径即：**

**ws.simple.exp.ExpenseReimbursementSystem**

**修改完毕后，保存备用**

# 2017年4月5日星期三（依据xsd校验xml是否符合规范）

我们使用javax.xml提供的API进行xml文档的校验，如果该文档满足我们定义的xsd规范，则返回true，否则返回false并且报告由于哪些元素违反了哪个规范

|  |
| --- |
| **package** cn.edu.ustb.xmlValidation;  **import** java.io.File;  **import** java.io.IOException;  **import** javax.xml.XMLConstants;  **import** javax.xml.transform.stream.StreamSource;  **import** javax.xml.validation.Schema;  **import** javax.xml.validation.SchemaFactory;  **import** javax.xml.validation.Validator;  **import** org.xml.sax.SAXException;  **public** **class** XMLValidation {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  File xsd = **new** File("E:\\ExFile\\exp.xsd");  File xml = **new** File("E:\\ExFile\\calculateReimbursementAmount.xml");  System.***out***.println("calculateReimbursementAmount.xml validates against exp.xsd? "+*validateXMLSchema*(xsd, xml));  }    **public** **static** **boolean** validateXMLSchema(File xsdPath, File xmlPath){  **try** {  SchemaFactory factory =  SchemaFactory.*newInstance*(XMLConstants.***W3C\_XML\_SCHEMA\_NS\_URI***);  Schema schema = factory.newSchema(xsdPath);  Validator validator = schema.newValidator();  validator.validate(**new** StreamSource(xmlPath));  } **catch** (IOException | SAXException |NullPointerException e) {  System.***out***.println("Exception: "+e.getMessage());  **return** **false**;  }  **return** **true**;  }  } |

验证代码如上所示

例子：xsd为exp系统的所有输入输出消息规范。Xml为calculateReimbursementAmount消息的输入数据。当输入数据为1时：（stafflevel满足枚举类型，actualmonthlymileage大于0）

|  |
| --- |
| <exp:calculateReimbursementAmount xmlns:exp="http://exp.simple.ws">  <exp:stafflevel>seniormanager</exp:stafflevel>  <exp:actualmonthlymileage>1.8</exp:actualmonthlymileage>  </exp:calculateReimbursementAmount> |

验证结果为true

当输入数据如下：（stafflevel不满足枚举类型，actualmonthlymileage大于0）

|  |
| --- |
| <exp:calculateReimbursementAmount xmlns:exp="http://exp.simple.ws">  <exp:stafflevel>JJT</exp:stafflevel>  <exp:actualmonthlymileage>1.8</exp:actualmonthlymileage>  </exp:calculateReimbursementAmount> |

验证结果为false并且给出了违反原因

|  |
| --- |
| Exception: cvc-enumeration-valid: Value **'jjt'** is not facet-valid with respect to enumeration **'[seniormanager, manager, supervisor]'**. It must be a value from the enumeration.  calculateReimbursementAmount.xml validates against exp.xsd? false |

当输入数据如下：（stafflevel满足枚举类型，actualmonthlymileage小于0）

|  |
| --- |
| <exp:calculateReimbursementAmount xmlns:exp="http://exp.simple.ws">  <exp:stafflevel>seniormanager</exp:stafflevel>  <exp:actualmonthlymileage>-1.8</exp:actualmonthlymileage>  </exp:calculateReimbursementAmount> |

返回结果为false

|  |
| --- |
| Exception: cvc-minInclusive-valid: Value '**-1.8**' is not facet-valid with **respect to minInclusive '0.0E1'** for type '#AnonType\_actualmonthlymileagecalculateReimbursementAmount'.  calculateReimbursementAmount.xml validates against exp.xsd? false |

**注：如果多个违反，也只提示第一个违反的= =，这个有点不大好我觉得。**

# 2017年4月2日进展的补充说明

**!!!使用原来创建AXIS web服务的方法也可以在网络访问的WSDL的内容中添加documentation以及约束标签。仅需要先正常发布web服务后，网络访问获取其wsdl保存，再扩展之后，再放入相应aar的META-INF\目录下即可。**

# 2017年4月7日星期五(xml数据生成工具总结)：

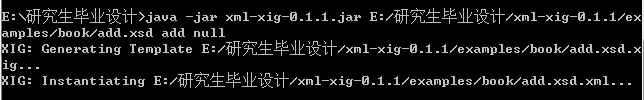
**注：所有使用的工具均放置在E:\xsd2xmlTool工具中！！！**

关于目前从xsd导出xml的已经找到的工具如下：

|  |  |
| --- | --- |
| 名称 | 是否开源 |
| Xig | 是 |
| CAMed | 否 |
| Xmlgen | 是 |
| DataGen | 无法下载 |
| XmlMate | 否 |

* **XIG**
  + 优点：开源有jar包可以使用，从Java代码中调用cmd即可使用；能够根据xsd生成xml框架；
  + 缺点：标签需要添加“xsd:”前缀！！！；不能分析出attributes、choice标签；当simpleType在元素外，不能分析到它的基本类型；原来工具文档中提到可以使用xig生成合法数据，但是xig标签解析目前已经不支持了（不能访问相关命名空间），因此该工具目前不支持生成合法数据。
  + 使用方式

**在xig的jar目录下**进行java -jar xml-xig-0.1.1.jar xsd文件 节点名 null 的执行，会生成xml框架 及其xig文件。



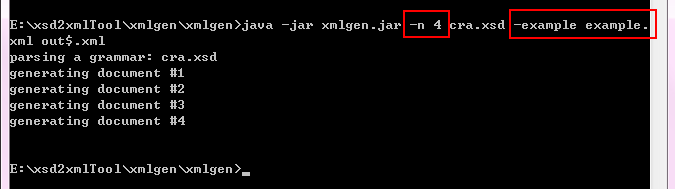
java -jar xml-xig-0.1.1.jar xsd文件 节点名 xig文件 生成有合法数据的xml

**注意**

* 用XIG分析的文档有一个问题：标签需要添加“xsd:”前缀！！！
* 不能分析出attributes、choice标签
* 当simpleType在元素外，不能分析到它的基本类型
  + 相关网站：http://xml-xig.sourceforge.net/
* XML Instance Generator（**Xmlgen**）**：**
  + **优点：能够生成xml有效数据，也可以生成无效测试数据（但是无效数据往往是缺少某个标签，而不是一个违反其约束的值）**
  + **缺点：对于xsd本身的约束支持不好，容易出现unable to generate value for this datatype: XXXX的错误，需要提供额外输入示例（不能处理的数据需要输入一个合法的值），之后生成的值都是用这个输入数据的！！！。**
  + **使用说明：**[**file:///E:/xsd2xmlTool/xmlgen/xmlgen/HowToUse.html**](file:///E:/xsd2xmlTool/xmlgen/xmlgen/HowToUse.html)
  + **使用：在xmlgen.Jar所在目录输入：**

**java -jar xmlgen.jar -n 100 myPattern.trex out$.xml**

**This will create out00.xml, out01.xml, ... out99.xml.**



* **CAMed (提供了可以直接使用的工具，但是无法用代码去调用，只能说是用代码打开它)**
  + **优点：可以生成一定数量的合法和不合法的xml数据**
  + **缺点：但是不合法的数据是随机生成的，并不符合我们定义的“一个无效区间覆盖一个”的准则，并且对于正则表达式的支持不是很完善（将正则表达式转换为自己的规则，但是转换过程可能有误）。**
  + **使用说明、**
    - **官方文档**[**https://wiki.oasis-open.org/cam/**](https://wiki.oasis-open.org/cam/)
    - **Generate XML test examples from XSD：**

[**https://wiki.oasis-open.org/cam/Generating\_XML\_test\_examples\_from\_XSD**](https://wiki.oasis-open.org/cam/Generating_XML_test_examples_from_XSD)

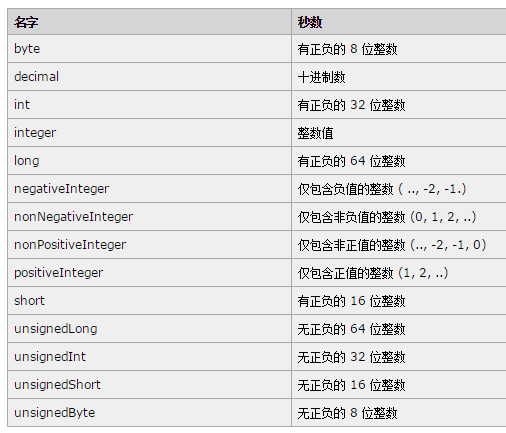
* + - **使用文档**[**https://www.oasis-open.org/committees/download.php/29661/XSD%20and%20jCAM%20tutorial.pdf**](https://www.oasis-open.org/committees/download.php/29661/XSD%20and%20jCAM%20tutorial.pdf)
  + **我们认为可以使用该工具，进行有效数据的生成即可，无效数据我们自己生成**
* **DataGen 官网如下：**[**http://iwm.uni-koblenz.de/datagen/download.html**](http://iwm.uni-koblenz.de/datagen/download.html)
* **XmlMate**
  + **没有官网及说明文档，无法使用进行数据的生成**
* **其他从github上找到的~明天再写吧~**

# 2017年4月25日星期二（XSD数据类型总结）

字符串类型：string、normalizedString、token

日期类型：date、time、dateTime、duration、gDay、gMonth、gMonthDay、gYear、gYearMonth 9个

数值类型：decimal、integer等14个





# 2017年4月26日星期三（使用PlantUML及Graphviz可视化显示模型）

* 安装plantuml：

主网址：<http://plantuml.com/>

首先下载<http://plantuml.com/download中的plantuml.jar以及语法说明文件PlantUML_Language_Reference_Guide.pdf>

将下载好的jar导入到Java工程中即可

* 安装Graphviz

除了时序图之外，一些其他的图，plantUML需要借助Graphviz

graphviz是贝尔实验室设计的一个开源的画图工具，从

<http://www.graphviz.org/Download_windows.php>

下载msi的安装版本，默认安装即可，因为plantUML要求

Windows :Firstly in: c:\Program Files\Graphviz\*\bin\dot.exe

Then in: c:\Program Files (x86)\Graphviz\*\bin\dot.exe

* 使用java调用plantUML

代码如E:\研究生毕业设计\plantUML中的ModelVisualization.java所示

# 2017年5月20日星期六（数据之间的约束测试用例生成）

关于数据之间的约束问题及测试用例生成：考虑使用因果图法及决策表+手工测试用例生成进行生成：

但是存在一个问题：现有的schema文档无法体现这个数据之间的约束问题，即即使是无效测试用例（即数据之间的组合是无效的），也能用过xml校验。

我们能够做到的只是说在最后测试失败后，能够给出可能由于无效组合，导致其发生错误

同时，传统的决策表针对的是一定选项后导致的动作（在知道程序内部逻辑值之后进行的设计，由于我们是黑盒测试，因此只知道程序规格说明，因此：），我们这里将动作定为输入合理与输入不合理

# 2017年5月27日星期六 和老师讨论后认为应该改进的（行为模型生成和测试用例生成相关方面）

* 行为模型中只显示有效测试序列即可，这个是服务的行为建模呀，当然是正确的了
* 测试用例生成放弃等价类划分，改用决策表方法，也许需要考虑生成测试用例的错误检测能力了
* 修改XSD即可，之后针对不同的xsd生成不同测试用例用例

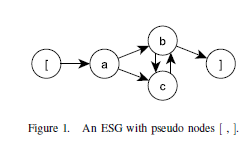
# 2017年5月31日星期三 事件序列图相关总结

**【Event- and Coverage-Based Testing of Web Services 2010】**

Belli [16]引入事件序列图ESG测试图形用户界面，使用简化方式描述用户交互，忽略被测试系统的内部状态ESG允许生成测试用例集（不仅仅是可取行为还有不可取行为）为待测程序提供全面的测试

此外，一个扩展的可拓学，命名为esg4ws [8]，用来进行web服务测试

ESGs用来描述有向图，节点代表事件，边缘表示事件的有效序列



在esg4ws，事件被指定为一个SU T的请求（一个操作调用）或响应（服务器响应消息到“请求”）。两个伪节点，[ 和 ]，象征进入和退出。任何从顶点[开始的和]结尾的序列被称为完整事件序列（CES），CES被用作正面测试输入

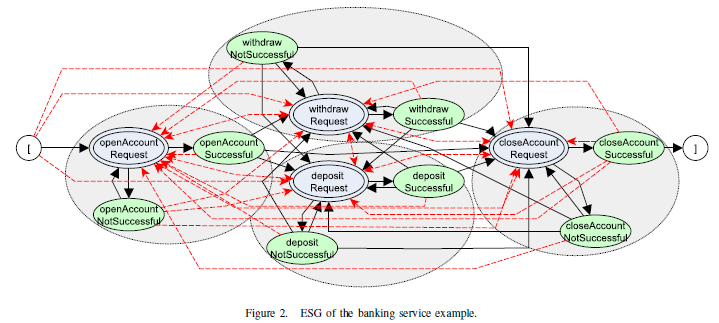
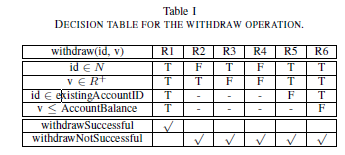


图2中属于同一操作的请求和响应被圆盘包围

表示操作的调用与ESG的节点输入参数细化的决策表（DT）



the WSDL does not describe valid or invalid sequences in which service operations are supposed to be called，因此无效测试也很有必要（为了测试系统的不良行为）

为使测试系统反应在一个意想不到的不良事件，FEPs将扩展到故障完整的事件序列

（fcess）。一个功能组别选举开始与伪顶点“[”考虑FEP结束。因此，否定测试系统，fcess作为测试输入。

**[Event-based modelling, analysis and testing of user interactions: approach and case study]**

Event-based otions and tools are used to generate and select test cases systematically.

基于事件概念和工具被用来系统的生成和选择测试用例

这个图形称为一个事件序列图（ESG）优点：

* 正确事件序列和意外事件序列的整体，这使得检查系统的行为是否符合用户的期望和故障处理得当

此外，从用户的角度来看，而与系统交互的设施。基本上，事件是外部观察到的现象，如环境或用户刺激，或系统的响应，强调不同阶段的系统活动。很显然，这样的代表无视系统内部具体的行为，ESG比状态转换图或有限状态机

ESG=(V,E)

* V：代表顶点（有限集合，不为空）
* E： 边的有限集合
* 入口节点集合和退出节点集合



* 一个事件可以是一个用户输入或一个系统的响应
* event sequence (ES) 事件序列：



Note that the pseudo vertices [ , ] are not included in the ESs

* complete event sequence, CES 完整事件序列：



# 2017年6月5日星期一 敲定事件序列图的定义与最后展示方式:

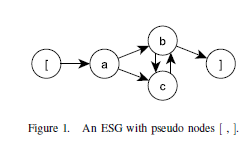
时间序列图的定义和文章：[Event-based modelling, analysis and testing of user interactions: approach and case study] 里面的定义是相同的！！！

使用该定义即可

Belli [16]引入事件序列图ESG测试图形用户界面，使用简化方式描述用户交互，忽略被测试系统的内部状态ESG允许生成测试用例集（不仅仅是可取行为还有不可取行为）为待测程序提供全面的测试

此外，一个扩展的可拓学，命名为esg4ws [8]，用来进行web服务测试

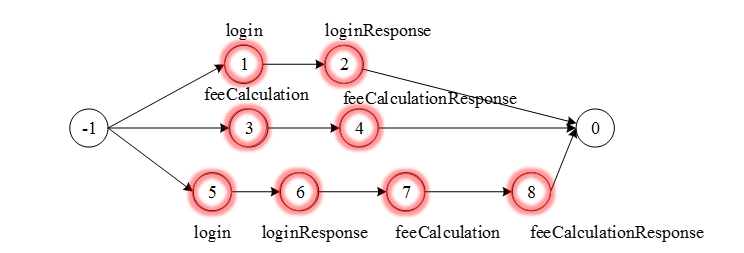
ESGs用来描述有向图，节点代表事件，边缘表示事件的有效序列



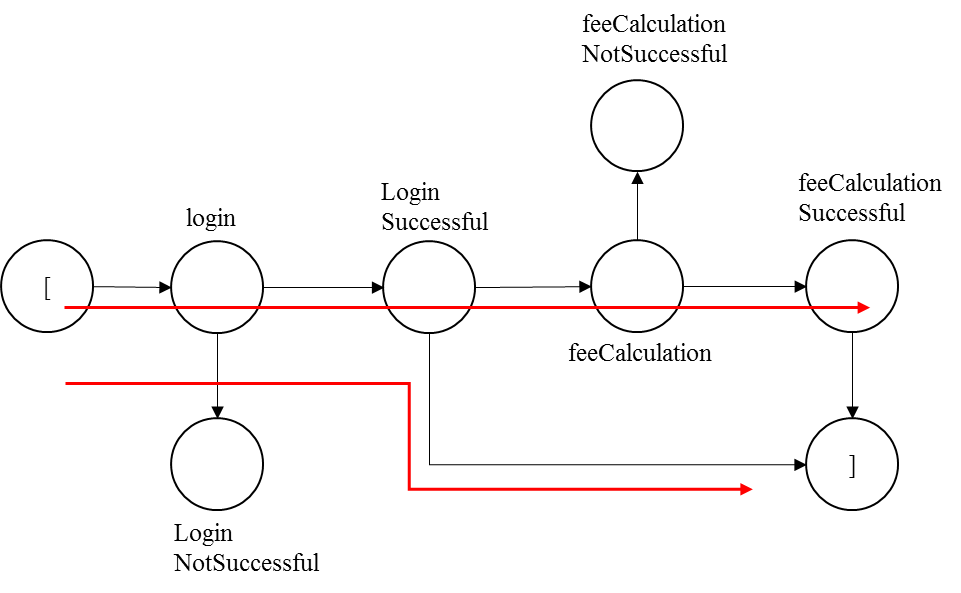
在esg4ws，事件被指定为一个SUT的请求（一个操作调用）或响应（服务器响应消息到“请求”）。两个伪节点，[ 和 ]，象征进入和退出。任何从顶点[开始的和]结尾的序列被称为完整事件序列（CES），CES被用作正面测试输入

我之前的老师说错误的原因是因为下面这个问题：如下图所示是我原来的模型图：

* 出现了一个login单独执行的边，又出现了login到feeCalculation，即这里把每一条路径展示出来了
* 其次单独执行feeCalculation路径是不正确的，不在事件序列图中展示



应该转换为如下图形，就对了：



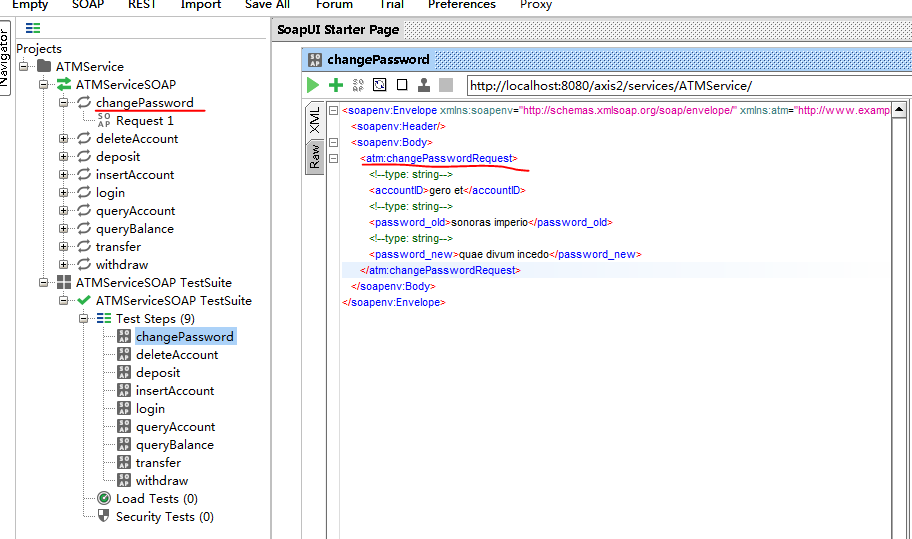
# 2017年6月7日星期三

SoapUI自动生成的测试框架生成的测试数据中的消息对应的名称为：相关WSDL中的xsd:schema标签里面的element名字

这个名字其实就是操作对应的消息对应的part!!!

我们在分析wsdl的时候得到的model中，使用的也是这个part的名字！！

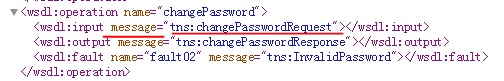
完全没有问题，对应的上的



Wsdl:







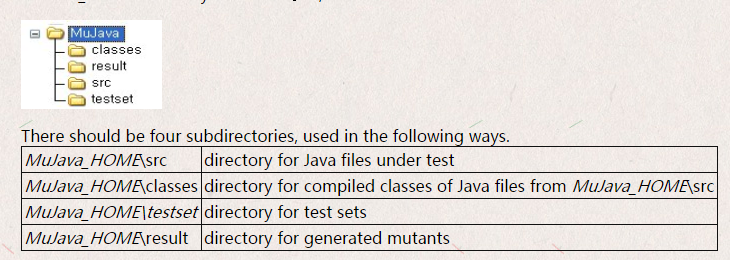
# 2017年6月12日星期一 mujava的安装与使用过程总结

Mujava官方网站：<http://cs.gmu.edu/~offutt/mujava/>

步骤：

* 下载mujava.jar、openjava.jar以及mujava.config
* 目录的建立：在指定的工作目录中建立如下结构,并将mujava.jar、openjava.jar以及mujava.config放入到该目录中. 我们建立的目录名称为E:\mujava

将mujava.config内容进行修改: 使得MuJava\_HOME=E:\mujava（自己建立的目录的绝对地址）



* 修改系统变量CLASSPATH，需要添加mujava.jar、openjava.jar、classes目录、junit库、java的tools.jar的路径，修改如下：

CLASSPATH=%CLASSPATH%;E:\mujava\mujava.jar;E:\mujava\openjava.jar;E:\mujava\classes;E:\junit\org.junit\_4.11.0.v201303080030;E:\junit\org.hamcrest.core\_1.3.0.v201303031735.jar;

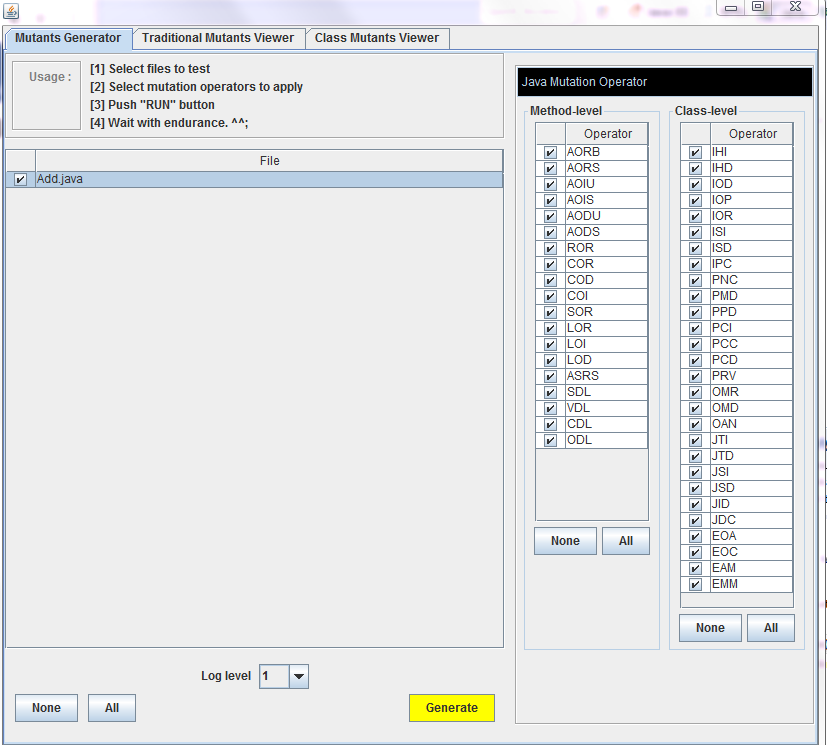
* 配置完成后启动cmd，使用**java mujava.gui.GenMutantsMain进行工具的使用。**

Mujava工具的使用

1. 生成变异体：

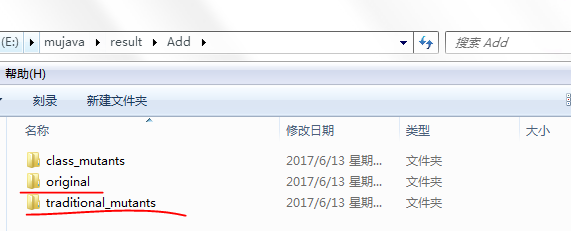
将想要变异的Java文件及其相关文件放入MuJava\_HOME\src，编译后的全部.class文件放入MuJava\_HOME\classes\。**java mujava.gui.GenMutantsMain进行工具的使用。**

**选择要变异的程序，以及方法和类上的变异体。点击Generate函数，生成所需变异体**

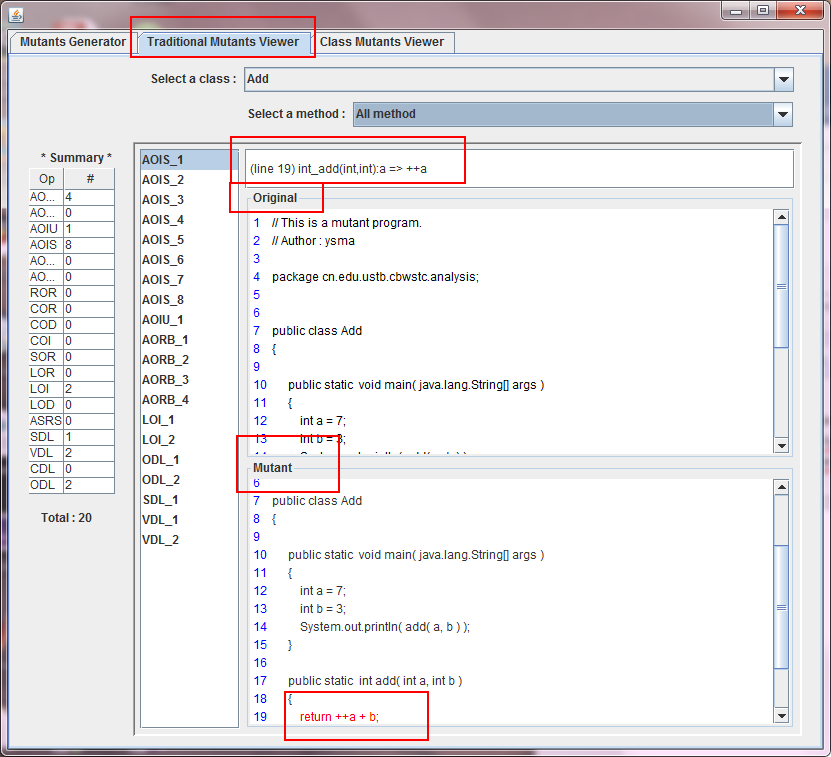


1. 查看变异体

生成的变异体被存放在MuJava\_HOME\ result\目录中。包括原始程序original文件夹（原始java以及class），使用方法层变异算子进行变异后产生的变异体及其class文件（放在traditional\_mutants文件夹里）



可以使用工具中的Traditional Mutants Viewer 界面查看结果，包括每个变异算子生成了多少变异体，每个变异体与源程序的变异的语句位置



1. 设计测试用例：mujava的测试用例设计为junit格式

# 2017年6月19日星期一 Graphviz

图形可视化是将结构信息表示为抽象图形和网络图。它在网络、生物信息学、软件工程、数据库和网页设计、机器学习和其他技术领域的视觉界面中具有重要的应用[17]。

Graphviz是开源图形可视化软件[1]。它有几个主要的布局器。它还具有Web和交互式图形界面，以及辅助工具，库和语言绑定。

Graphviz布局程序以简单的文本语言描述图形，并以有用的格式制作图表，如网页的图像和SVG; PDF或Postscript纳入其他文件;或显示在交互式图形浏览器中。

Graphviz对于具体的图表有许多有用的功能，例如颜色，字体，表格节点布局，线条样式，超链接和自定义形状的选项。

在实践中，图形通常是从外部数据源生成的，但是也可以手动创建和编辑图形，作为原始文本文件或图形编辑器。

Graphviz中含有一下的一些布局器：

（1）dot：分层或者有向图的分层图。如果边缘具有方向性，这是使用的默认工具。点的方向是从上到下或从左到右，同时会尝试避免边缘交叉并减少边缘长度；我们这里就是使用dot。

（2）twopi：径向布局；

（3）circo：圆环布局；

（4）fdp：用于包括处理较大图形和集群无向图的多网格求解器；

（5）neato：“spring model”布局。如果图形不是太大那么这是一个好的布局器。

Graphviz设计的最初目的是对有向图或者无向图等进行自动布局。

首先，在dot脚本中对图的顶点和边进行定义，其中顶点和边都含有各自的属性，例如颜色，形状，字体，样式，填充模式等。然后在布局中运用适当的算法布局。布局算法在绘制各个顶点和边的时候，需要让各个顶点尽可能的在画布上均匀的分布，同时尽可能的减少边与边之间的交叉(当边与边之间存在过多的交叉时，很难去区分辨别顶点之间的关系)

Graphviz生成的文件后缀名为.gv。每个后缀名是.gv的文件都代表一个图，我们可以选择通过命令行的方式生成图像，或者可以使用gvedit，Graphviz提供的工具来编辑和运行脚本。

[1]严代彪, 王树宗. 一种源程序到程序流程图的自动生成算法[J]. 微计算机信息, 2003, 20(07): 83-65.

[17]Mostafijur Rahman,Wendy MacCaull. An Application Suite for Service Enabled Workflow[J]. Procedia Computer Science, 2016.

[19]张伟欣. 基于Graphviz的ProM模式提取插件图形系统的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2015.

例子： 安装并配置完Graphviz之后，使用程序将生成的graph对象转换为Graphviz语言，使用dot -Tpng Graphviz.txt -o Graphviz.png进行可视化转换

解析扩展后的WSDL，

* 解析结果被存储在Parser文件夹的result中
* mapXI存储了所有操作的输入数据格式，并将其赋值给ParsePanel类中的map
* mapC存储了操作对应的约束描述格式为Map<Opr,Constraint> 没有约束value为NullDocumentation
* mapIO存储了所有操作的调用响应part 格式为Map<Opr,OprI#OprO>
* mapSq存储了执行一个操作需要的序列格式为Map<Sequence,Opr>

graph由nodes和edges组成，其中nodes是LinkedHashSet（为了防止相同node被多次添加进图中）edges是ArrayList类型的

node类由id、name、type、condition、beforeNodes、afterNodes组成

id、name、type、condition为了标识节点

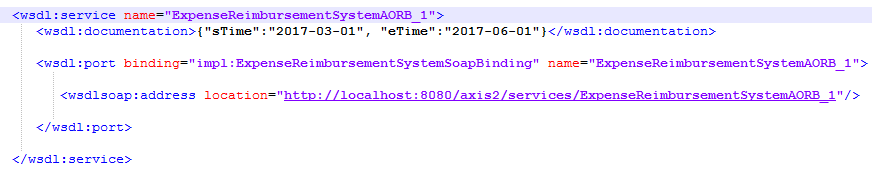
beforeNodes、afterNodes为了后期生成边（两个都是LinkedHashSet，防止被重复添加）

Edge类由from和to组成，为了根据node的id将节点连接起来

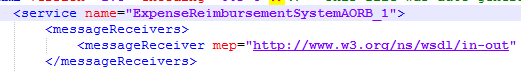
# 2017年6月29日星期四 变异测试执行过程中关于变异体部署问题：

* 每个变异体单独部署为一个Web服务，修改services.xml以及wsdl即可

其中wsdl需要修改service name以及 location



Services.xml仅需修改name即可，其余东西不需要变换



复制n多个aar，把WSDL、class文件以及Services.xml替换，重启服务器

* 只需要一个web服务，每次测试用例执行不同服务时。进行class的替换、服务器重启执行，这个问题就是切换代价太大，等待时间，内存消耗，不太提倡
* 在进行变异测试时，无需在web服务环境下进行，将待测程序与变异体看做简单的java程序运行即可；
* 比较倾向于1、3两条

最后决定的是使用第一个~

# 2017年7月10日星期一 模型覆盖准则：

* Operation覆盖：覆盖全部web服务提供的操作
* Message/Node覆盖：覆盖全部web操作的消息
* edge覆盖：覆盖全部转移边
* edge+DT覆盖：针对相同的转移边，考虑内部状态，进行覆盖

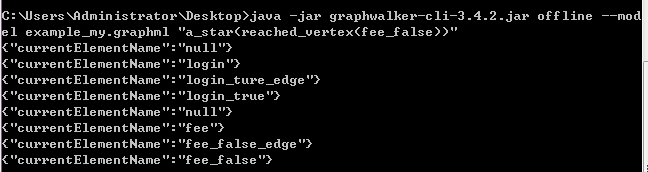
使用**graphwalker开源工具，针对不同覆盖准则进行测试序列的生成**

（<http://graphwalker.github.io>）

模型转换为符合该工具语法的graphml文件

之后使用以下命令进行测试序列的生成：

* java -jar graphwalker-cli-3.4.2.jar offline --model example\_1.graphml " a\_star(reached\_vertex(v\_Browse))" 覆盖节点
* java -jar graphwalker-cli-3.4.2.jar offline --model example\_1.graphml " a\_star(reached\_ reached\_edge(v\_Browse))" 覆盖边



注意。这个图的Start节点只能有一个出度，不然会使得遍历出现问题。

因此我们把Start这个节点添加到我们自己图中的节点里，并且加一个init边即可（如果测试不需要初始话的话，就加null的边）

# 2017年7月20日星期四

可以在论文中写到的：

* 扩展的WSDL中约束的形式化定义
* 扩展的事件序列图的定义
* 扩展WSDL转换为事件序列图的算法
* 覆盖准则的定义
* 测试序列生成及去冗余算法（针对不同测试准则算法可能有所不同）
* 去冗余过程中定义什么是“序列长度”“序列包含”
* 待加入

WSDL-Based Automated Test Data Generation for Web Service

首先对于操作的输入元素数据类型定义了形式化的模型，其次，本文提出了一种从WSDL中导出模型的算法，第三，提出了一种针对模型中的操作生成测试数据的方法。

代理和用户测试Web服务包括两个不同的级别：

1）测试Web服务的单一操作；

2）测试Web服务的操作顺序。

WSDL is a XML-based language to define Web Services WSDL是一种基于XML的定义Web服务的语言，它指定服务的位置以及服务公开的操作或方法。

WSDL规范的XML表示形式如下所示：

<**description**>

<documentation />?

[ <import /> | <include /> ]\*

<types />?

[ <message /> | <portType /> | <binding /> | <service /> ]\*

</**description**>

# 2017年7月22日星期六 XML合法性验证问题

使用SOAP以及WSDL中的schema标签及其内容。注意事项

* 不可使用WSDL全文
* 使用schema时，其中targetNamespace不能重复 还要加入其它命名空间变量
* 即使输入数据有限制标签，不添加内容也是可以的~~

当xsd xml有中文的时候，请使用utf-8对其进行读取，存储

InputStream ifile = **new** FileInputStream(xmlPath);

InputStreamReader ir = **new** InputStreamReader(ifile, "UTF-8");

Xml必须取soapbody中的内容，加入xmlns:xxx="xxx" 命名空间 才能进行验证

# 2017年7月25日星期二 关于使用cmd运行有import的java文件

先cd到需要编译执行的Java文件的所在目录

之后运行如下命令：

|  |
| --- |
| 编译 |
| javac –classpath %classpath%;需要的jar的路径 待编译的Java文件 |
| 运行 |
| java –classpath %classpath%;需要的jar的路径 待执行的class文件 |

# 2017年7月27日星期四 测试序列编程框架的一些记录

TestSequenceModel类 包含一系列TeatSequence ，是TeatSequence的集合

一个测试序列是一个TeatSequence对象，TeatSequence包含如下信息

* private int tcId; //测试序列编号
* private ArrayList<String> coverElement; //覆盖对象集合
* private String ts; //测试序列
* private int tsLength; //测试序列长度

CoverageCriteria类表述了四种测试用例覆盖准则

GetInitialTestSequence 生成测试序列

# 2017年7月28日星期五 约束求解代码问题:

* Int类型初始化：
* IntVar v = model.intVar("v", IntVar.MIN\_INT\_BOUND, IntVar.MAX\_INT\_BOUND);
* boolean类型初始化
* BoolVar v = model.boolVar("v");

其他使用model.arithm进行匹配即可

# 2017年7月31日星期一 z3约束求解

Z3的安装请参考文件：E:\研究生毕业设计\z3安装教程\Z3约束求解安装说明

之后使用Java调用cmd执行如下语言

z3 约束文件.txt

其中约束文件里面写着SMT语法格式：

|  |
| --- |
| (declare-const a Int)  (declare-const b Int)  (declare-const c Int)  (declare-const d Real)  (declare-const e Real)  (assert (> a (+ b 2)))  (assert (= a (+ (\* 2 c) 10)))  (assert (<= (+ c b) 1000))  (assert (>= d e))  (check-sat)  (get-value (a))  (get-value (b))  (get-value (c))  (get-value (d))  (get-value (e)) |

String equality (= X "abc")

String variable (declare-const X String)

(declare-const b String)

(declare-const a String)

(assert(= b "abc"))

(assert(= a b))

(declare-const b String)

(declare-const a String)

(assert(= b "abc"))

(assert(= false (= a b)))

(check-sat)

(get-value (a))

(get-value (b))

(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"))

# 2017年8月3日星期四 错误出现时定位到底由于什么错误引起的：

|  |
| --- |
| 1. 执行前测试前查看服务可用时间约束，如果现有时间大于可用时间，提示**可能违反时效约束**，服务可能存在由于更新或停止服务等不可用的风险。 2. 执行前查看客户端IP地址是否违背测试序列中的服务区域约束，如果违背提示违反哪个操作的**服务区域约束，结束本次执行；** 3. 执行调用前，查看测试脚本中SOAP消息是否通过XSD校验，若未通过则提示违反了哪个操作的**服务输入参数范围约束，结束本次执行；** 4. 执行测试过程中，查看每次调用的测试输出是否违反XSD，如果违反则进行后续校验步骤（5、6、7、8、9），否则继续执行测试； 5. 查看测试序列是否满足被测操作的执行顺序约束，如果不满足，提示违反了哪个操作的**序列约束（顺序），**测试结束； 6. 查看测试序列是否违反了被测操作的重复调用约束，如果违反，则提示违反了哪个操作的**序列约束（重复调用），**测试结束； 7. 查看测试数据是否满足序列上的所有参数关系约束，如果违反则提示违反了**参数关系约束，测试结束** 8. 查看所调用操作是否存在调用操作，如果有，提示可能由执行该操作的调用操作出错，测试结束； 9. 若上述约束均为违反，提示测试人员违反了其他未知错误，测试结束； 10. 测试用例通过，提示该条测试用例执行成功。 |

1. 测试前查看服务可用时间约束，如果现有时间大于可用时间，提示可能违反时效约束，服务可能存在由于更新或停止服务等不可用的风险。
2. 查看测试脚本中的测试序列是否有违反ipconfig的 如果有提示违反了服务区域约束
3. 查看测试脚本中soap是否符合xsd 如果不符合，则提示违反了服务输入参数范围约束
4. 前三步没有违反的话，看测试输出是否存在“”字样如果存在，提示服务不可用。（专门做一个变异体来支持它）
5. 前四步都没有问题，看测试输出是否违反xsd，如果违反了可能存在 违反服务序列约束、调用约束、参数关系约束三个约束的问题。
6. 首先查看是否违反了序列约束，看调用的操作，是不是有执行顺序约束以及重复执行约束，如果都没有就不用管了，如果有的话，就记录有约束的操作调用前的序列，看看是不是违反了调用序列。
7. 如果没有违反序列约束，就判断所执行的脚本中的操作是否调用了其他操作，如果有，提示可能违反调用约束
8. 如果都没有，提示可能违反了参数关系约束

# 2017年8月8日星期二 程序分析

GetInitialTestSequence 用来按照选择的覆盖准则以及之前生成的行为模型图进行测试序列的生成，包括初始测试序列的生成以及去除冗余后测试序列的生成

变量：

* String jarPath graphwalker-cli-3.4.2.jar所在路径
* String graphFile 符合graphwalker的graph类型文件，用来进行模型遍历
* ArrayList<String> initTs 存储初始测试序列
* HashMap<String,Integer> initTsmap 存储初始测试序列 其中为序列字符串及序列长度，为去冗余做准备

chooseCovCri 选择覆盖准则方法

getInitGraphWalkerResult 生成初始测试用例 放到initTsmap中

reRedundancy(initTsmap,ele) 进行冗余删除 返回TestSequenceModel类型的tsm变量

TestSequenceModel 由ArrayList<TeatSequence> TeatSequence类型的一堆序列集合组成

TeatSequence 类 有如下属性

* private int tcId; //测试序列编号
* private ArrayList<String> coverElement; //覆盖对象集合
* private String ts; //测试序列
* private int tsLength; //测试序列长度

# 2017年8月9日星期三 测试脚本设置：

<Sequence>

<Operation name=””id=””>\*

</Operation>

</Sequence>

DataToCase需要做的事情：

* 方法1：获取约束求解后所有变量的值，保存在一个hashmap中。
* 方法2：分析测试序列，写测试脚本
* 方法3：在测试脚本中进行数值的替换

# 2017年8月16日星期三 工具使用

进行解析后，XSD中的文件要添加多个命名空间

之后xml中的文件需要进行命名空间的移动

上述两个步骤要进行，之后才能继续

关于违规测试序列的生成： 传递的是graph

* 遍历其中的input的node 之后如果说该node的IterationC约束为false 则生成一个重复调用该操作的序列
* 如果该node的PreOpC约束不为nullCondition，则生成一个直接调用该操作的序列。

NoSuchMethodError

Class Not found : org.gjt.mm.mysql.Driver

Graph图的生成

* Graph名字为wsdl中抽取的名字
* Id=0, name=Strat, type=INITIAL
* Id=1, name=Init, type=INITIAL
* Strat后继为Init，Init前继为Strat
* 读取之前分析WSDL之后生成的result.txt
* 读取第一行，graph的endtime设置
* 之后循环读取每一行，针对每一行进行如下设置：
  + 读取split[1] nodeI name= split[1] type=input 参数约束：使用split[0]找到mapXI中对应的约束，进行添加, 将其他约束：分析split[3]里面的所有约束，进行添加
  + 如果nodeI没有序列约束，那么nodeInit的后继为nodeI，nodeI的前驱为Init
  + 接着读取split[2] nodeO1 name= split[2]+\_succ type为output
  + NodeI 的后继为nodeO1 nodeO1的前驱为nodeI
  + 如果nodeI的重复调用为true 那么nodeO1的后继为NodeI NodeI的前驱为nodeO1
  + split[2] nodeO2 name= split[2]+\_fail type为output
  + NodeI 的后继为nodeO2 nodeO2的前驱为nodeI
  + Node end 先不设置id，type类型为end 将nodeO1与nodeO2后继为Node end并且Node end 的前驱为nodeO1 nodeO2
* 循环完成后，将node end的id设置上即可
* 循环所有已经存在的node 找出类型为input并且序列id不为空的：针对每一个oknode进行如下循环：
  + 读取oknode的序列约束
  + 人工界面进行添加（暂时的）

|  |  |
| --- | --- |
| Algorithm 1 解析EX-WSDL生成服务行为模型算法 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | 初始化一个行为模型g（Nodes=Ø、Edges=Ø），将其名称属性设置为EX-WSDL URI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节点及Init节点，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  解析EX-WSDL，获取其全部操作集合OpSet  **for** each op = element in OpSet **do**  创建opReq节点，设置节点类型为input，解析op操作的paraRange、ipRegion添加进opReq节点约束属性中  创建opRes\_succ节点，设置节点类型为output，设opReq为opRes\_succ前继节点；opRes\_succ为opReq后续节点  创建opRes\_fail节点，设置节点类型为output，设opReq为opRes\_fail前继节点；opRes\_fail为opReq后续节点  向Nodes中添加opReq及opRes节点  **if** opIteration == true **then**  设置opRes\_succ为自己的前继及后续节点  **end if**  **if** op preOp == null **then**  opReq添加进Init的后续节点，Init添加进opReq前继节点  **else**  解析preOp约束，将有前后关系的节点进行关联  **end if**  **end for**  **for** each node in Nodes **do**  **for** afterNode in node. afterNodes **do**  Add edge to Edges  **end for**  **end for** |

# 2017年8月28日星期一 关于违反序列约束的测试用力生成问题

针对违反序列约束的测试序列 只需要生成符合参数范围约束的测试数据即可

Pattern与 enumeration 转换为正则求解

其他数据则需：确定一个数的范围和类型 之后随机生成数据

目前存在的问题：

针对序列negative（Iteration）类型的测试用例生成： 重复调用的操作的数据必须是一致的！！！(解决了)

# 2017年9月2日星期六 违反约束的定位过程：

如果出错了：

* 首先判断是否违反了重复调用约束：针对有所有调用约束的input node 查看测试序列是否存在nodeNamerequest\_succ#ex(为一个字符)# nodeName这样的调用。
* 如果有，则提示违反了nodeName的重复调用约束
* 如果没有，则判断是否违反了序列约束，找到所有input node中有序列约束的，之后看测试序列nodeName前面的序列是否符合其正则表达式，如果不符合，则为序列约束
* 如果前面都符合，判定为参数关系约束