用户需求驱动的 Web 服务测试

许 蕾 陈 林 徐宝文
(南京大学计算机科学与技术系 南京 210093)
(南京大学计算机软件新技术国家重点实验室 南京 210093)

摘 要 Web 服务涉及开发者、提供者、注册中心、用户等多方,各方测试目的、所掌握的资源以及所使用的测试方法各异.以用户需求来驱动 Web 服务测试,更切合现状且能提升测试效率.文中建立了用户需求特征模型,引入层次关系、控制结构、约束条件等要素;对照目标级、服务组合、原子服务需求,分别进行基于等价类划分、数据流分析和变异测试的测试用例生成选择;当需求变动时,通过波动分析能迅速定位到服务的相关路径和变量,保证回归测试的效率;最后通过行程安排实例展示了工作流程和实验结果.

关键词 Web 服务测试;用户需求;测试用例生成;依赖性分析;回归测试
中图法分类号 TP311 **DOI** 号: 10.3724/SP.J.1016.2011.01029

Testing Web Services Based on User Requirements

XU Lei CHEN Lin XU Bao-Wen
(*Department of Computer Science and Technology, Nanjing University, Nanjing* 210093)
(*State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing* 210093)

Abstract Web service testing can be carried out by developers, providers, integrators, third-party certifiers and users, and different testers have different testing objects, testing resources and testing techniques. In realities, testing Web services in the view of users is more practical than others, and can improve testing efficiency in the case study. The authors set up user requirement model firstly, which began with general and abstract object requirements, and then decomposed and refined into concrete and determined atomic Web services, together with such key elements as layer relationships, control structures and restrict conditions, presented as a tree-style model; Next, test case generation and selection is carried out for the object level, service combination level and atomic service level of the user requirement model, and the corresponding techniques are based on equivalence division, data flow analysis and mutation testing; Then, the regression testing would have better efficiency. Based on the dependence relationships in the user requirement model and the fluctuation analysis method, the related paths and variables of Services under testing could be determined quickly; Finally, the workflow and case study are shown with the travel scheduling example.

Keywords Web service testing; user requirement; test case generation; dependence analysis; regression testing

收稿日期: 2010-11-08; 最终修改稿收到日期: 2011-05-23. 本课题得到国家"九七三"重点基础研究发展规划项目基金(2009CB320703)、国家自然科学基金(60873050, 90818027, 61003020, 91018005)、武汉大学软件工程国家重点实验室开放基金(SKLSE20080717)资助.
许 蕾, 女, 1978 年生, 博士, 副教授, 主要研究方向为 Web 服务分析测试. 陈 林, 男, 1979 年生, 博士, 主要研究方向为软件分析、软件重构. 徐宝文(通信作者), 男, 1961 年生, 博士, 教授, 博士生导师, 中国计算机学会高级会员, 主要研究领域为程序设计语言、软件工程、并行与网络软件. E-mail: bwxu@nju.edu.cn.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

1 引 言

Web 服务致力于建立一个可互操作的分布式应用程序平台，从而对基于不同开发语言、部署平台、通信协议的系统间有效交互以及在数据更新同步、信息一致性等问题上发挥作用. 理论上，Web 服务的互操作性有助于提高软件的重用性，但是，当用户有众多服务可选时，由于不足以信赖这些服务或多个服务组合时产生了问题，反而会影响到 Web 服务的发展和应用. 因此，实现真正高可信的 Web 服务是一项迫切而艰难的任务，需要从功能、性能、安全性等多个方面进行考虑[1]，并且研究视角也有 Web 服务开发者、提供者、集成者、第三方确认者和用户之分[2-3].

软件测试是保障软件质量的重要方法. 从 Web 服务的使用者(用户)角度来看，现阶段 Web 服务测试的主要挑战包括：对单个 Web 服务来说，由于用户通常只能获得 Web 服务的接口信息(WSDL 文档)[4]，因而只能进行黑盒测试，虽然目前有一些研究对 WSDL 文档进行了扩展，增加了一些运行时信息、语义信息或者约束条件[5]，但由于无法获取到程序代码，还是无法开展严格意义上的白盒测试；对单个或多个服务的运行来说，运行环境具有未知性；在 Web 服务的实际运行过程中，由于网络连接的故障或者服务提供者自身的原因，会导致某些服务暂时不可用，需要临时更换功能类似但环境不同的新服务，如果事先没有测试过，通常会带来很多问题；另外，Web 服务的分布式和并发执行[6]特性也增加了测试的困难和复杂性，有些场景难以重现和回放，定位故障和排错的难度很大.

由于用户是 Web 服务的使用者、评价者和最终决定者，本文以用户的需求为 Web 测试的驱动，在现有工作的基础上，充分挖掘用户的需求，并建立从需求到实现的桥梁，从而给出更有针对性和实施效率的测试策略. 分以下几个部分展开：首先，尽可能客观、准确地描述用户需求，分层次、多维度地进行刻画，使得模型能真正反映用户的需求并有一定的可扩展性，拟采用用户需求特征树来表达；其次，根据用户需求模型组织、实施测试，对照每一条需求，进行测试用例的生成和选择，即只覆盖用户关注的语句或路径；另外，当用户需求发生变动时，能迅速定位到服务的相关路径、元素和变量，从而保证全面、有效的回归测试；最后，将通过一个实例系统地展现建模、测试方法等方面的工作.

2 相关工作

Web 服务的出现解决了不同平台、不同编程语言以及异构的软件应用之间的互操作问题，要使 Web 服务得到进一步的发展和更广泛的应用，需要运用验证、测试技术来保障 Web 服务的正确有效运行. 形式化验证技术从服务提供的描述文件入手，提取功能或者属性规约，利用特定的建模技术对其实施形式化，并将形式化模型转化为检验工具可以接受的语言，最后用模型检验工具验证模型的正确性[7]. 而 Web 服务测试方面的相关工作，可以从测试视角和测试技术两方面来总结、分析，具体表述如下：

Web 服务从开发到运行，涉及开发者、提供者、集成者、UDDI 注册中心和用户等多方面因素，由于环境、着眼点等差异，各方面也均有各自不同的测试视角[2-3]. Web 服务的开发者可以利用白盒技术检测程序中存在的错误，而其他人员大多只能用黑盒技术来进行测试：服务提供者需要确保服务能够符合事先和服务请求者协商的约定；服务集成者需要确保任何绑定到组合应用中的服务都能满足其功能性和非功能性的要求；UDDI 注册中心需要对注册服务的质量进行评估，确保已注册的服务均有一定的质量保证且能够向服务请求者提供最优的满足其需求的服务供应商；用户只关注在自己的应用中所使用的服务，能保证正常工作即可，但用户通常难以精确地描述自己的需求，且这个需求会经常变动，如何从用户的角度出发来开展测试，是一个需要深入探讨的研究问题.

在针对单个 Web 服务的功能测试方面，通常是应用传统软件测试的各种技术，即将单个 Web 服务当作一个功能模块来处理[4]. 但 Web 服务还包括服务的描述、发布、发现以及消息的传输等内容，相应的还需要更多的保障措施. 有研究者提出进一步扩展 WSDL 语言，以增强其描述能力，为 Web 服务测试提供更多的信息，如增加前置、后置条件等[8]，或者引入语义本体的概念，更精确地描述服务接口及其关联、约束[5]. 相应地，在测试用例生成选择技术方面，由于测试者无法得到服务源码，通常是根据 WSDL 文档中的信息生成测试用例，主要采取随机法与边界值选取数据，操作简单但精度很低，不利于错误的发现. 在此基础上，研究者提出了基于本体论的 Web 服务测试用例生成技术[5]，通过 Ontology 推理得到测试数据；或者通过扩展的 WSDL 文档、

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

OWL-S 规格说明自动生成测试用例[8]. 自动生成的测试用例数量众多、质量良莠不齐, 通常采用变异测试方法来评价、筛选测试用例: 文献[9]提出了基于合约变异的 Web 服务测试技术; 文献[10]提出了一种基于 OWL-S 需求模型的 Web 服务变异测试方法; 文献[11]对 OWL-S 进行变异, 包括数据变异、条件变异、控制流变异和数据流变异.

从研究现状来看, 针对 Web 服务的测试研究还有一些不足. 例如, 现有的测试用例生成技术大多从 WSDL 入手, 利用传统软件测试的方法产生测试数据, 通常局限于单个服务; 对组合的 Web 服务测试还处在初级阶段, 主要都是基于分析 BPEL 和 OWL-S 文档的流程结构进行测试或采用模型检查的方法进行验证, 难以测试运行时动态组合的 Web 服务. 因此, Web 服务测试还需要进一步的研究和探索, 努力尝试新的测试视角和测试技术, 例如, 从用户的角度出发, 采用需求驱动的方式来进行测试. 而在需求驱动的测试用例集约简方面, 我们有一定的工作基础[12-13]: 首先充分挖掘并使用测试需求的相关信息建立测试需求的关系模型, 再定义测试需求间的独立、相交、包含和等价关系, 并提出了精简测试需求集的概念, 采用各精简算法最终实现测试用例集约简; 然后将此思想应用到基于组件的 Web 应用测试中, 基于各种覆盖率准则, 并通过分析等价和包含关系, 约简测试需求集合. 借鉴此思想, 可实施需求驱动的 Web 服务测试.

3　用户需求建模

3.1　用户需求的特点和要素

用户使用搜索引擎时, 首先需要组织好搜索关键词, 以便表达自己的需求. 但由于用户不够专业, 通常会词不达意. 类似的, Web 服务的用户对其究竟需要什么样的服务也往往不够明确, 通常会有一个大概的目标, 但具体需求含糊不清且可能会经常变动. 因此, 要对用户需求建模, 首先需要定性分析用户的需求, 提炼出用户需求的基本要素; 进而建立用户需求特征树型结构图, 即从用户总体需求出发, 逐层分解、逐步细化, 每一层均有其对应的目标、前驱后置条件、输入、输出、QoS 要求等内容, 分划的层数由实际情况来确定, 直到最终的叶子节点可以与某个原子服务相匹配.

目标是第一要素, 是用户要做某事的原因, 这直接决定了所需 Web 服务的类型、功能等内容, 通常在匹配时对应于 Web 服务的名称, 但由于大多数用户表述不够精确, 而且有近义词、多义词的影响, 会带来一些匹配上的问题.

前驱后置条件是用户需求的上下文环境, 这会影响用户的目标、QoS 要求和输出, 有时候会对 Web 服务产生非常大的影响, 因为环境变化了, 有些服务可能会运行不了, 从而导致故障的产生.

输入 *Input* 是 Web 服务运行前需要用户提供的信息, 对应于 Web 服务中 WSDL 文档中有类型要求的若干输入变量. 这需要 Web 服务在表示时尽可能地符合用户的使用经验和习惯.

同样, 输出 *Output* 也是 Web 服务运行前需要用户提供的信息, 对应于 Web 服务中 WSDL 文档中有类型要求的若干输出变量. 这也需要 Web 服务的输出能够充分体现用户的期望.

QoS 要求是用户非功能需求的体现, 表现在性能、安全性、可靠性、可用性等方面, 用户有 QoS 需求, 但在可接受程度上差异较大, 不同用户的 QoS 要求差别也很大, 最好能提供一些组合方案供用户来选择.

3.2　用户需求模型

定义 **1**.　用户需求模型 $M=(N, PN, QN, R, C, F)$是一个六元组, 其中

$N=\{ARN, DRN, SRN\}$, ARN(Atomic Requirement Nodes)是由一个原子服务所对应的需求用例对象; DRN(Description Requirement Nodes)是用户描述需求节点, 没有原子服务与其对应, 通过精化关系生成新的 DRN 或 ARN; SRN(Structure Requirement Nodes)是虚拟的节点集合, 用于描述各个 ARN 和 DRN 之间的结构化关系, 由多个 ARN、DRN 或 SRN 通过一定关系(M 中的 R 关系)组合成的结构化需求描述对象;

$PN=\{\text{Attribute of } N \mid IN, OUT, OBJECT, \text{etc.}\}$, PN 为节点 N 的功能性属性, 包括输入、输出、目标等内容, PN 是 N 的可选项;

$QN=\{\text{Quality of } N \mid ResponseTime, Usability, Reliability, \text{etc.}\}$, QN 为节点 N 的非功能性属性(QoS 属性), 包括响应时间(从用户发出请求到用户接收到服务的时间, 是评价服务客观性能高低的重要属性)、可用性(衡量一个服务持续可用的能力, 一个服务在一个观察时段 T 内, 其中有 t 时间可用, 则可用性可以表示为 t/T)、可靠性(综合效果评价, 分高、较高、中、较差、差等几个级别)等, QN 是 N 的可选项;

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

$R=\{Refine, Sequence, Split, Split\text{-}Join, Choice, If\text{-}Then\text{-}Else, Repeat\text{-}While, Repeat\text{-}Until, Concurrence\}$，该集合中的每个元素代表了节点之间的各种关系，如 *Refine* 表示 *DRN* 节点间、*DRN* 与 *ARN* 节点间的精化活动，另外的 *R* 关系表示 *SRN* 节点间、*SRN* 与 *ARN* 节点间的各种结构化活动；

$C=\{Control\ relationship\}$，是需求节点 *N* 之间关系的一种约束，例如需要满足一定的约束条件后才能建立两个节点的分支关系；

F 是节点关系的映射函数，定义为 $C: N\times N\rightarrow R$，即任意两个节点之间满足约束 *C* 的关系.

为便于说明，结合行程安排服务(Travel Service)实例来加以解释.行程安排服务是一种应用比较广泛的服务组合流程，它根据用户要求选择合适的航班(或车次)，并预订旅馆客房.要实现这一功能，行程安排服务从根节点出发，先细化为交通工具预订(vehicle)和旅馆预订(lodge)这两项子需求，再进一步细化为航空预订服务(AS)、火车预订服务(TS)、旅馆预订服务 1(H1)、旅馆预订服务 2(H2)这四项子需求，这四项子需求有对应的原子服务，此时停止细化，完成用户需求的功能划分.这些节点可以有各自的 *PN* 和 *QN* 属性，上层节点与下层节点之间存在精化的关系(Refine).另外，在这个实例中，要求这些原子服务可并发执行，交通工具预订(vehicle)和旅馆预订(lodge)也需要并发执行，为了描述这些结构化关系，我们引入了 *SRN* 节点和对应的 *R*(Concurrence)边(如果用户需要交通工具预订和旅馆预订顺序执行，也可以给出另外的 *SRN* 和 *R*(Sequence)边.据此建立的需求模型如图 1 所示，图例说明在图 1 的右下角.

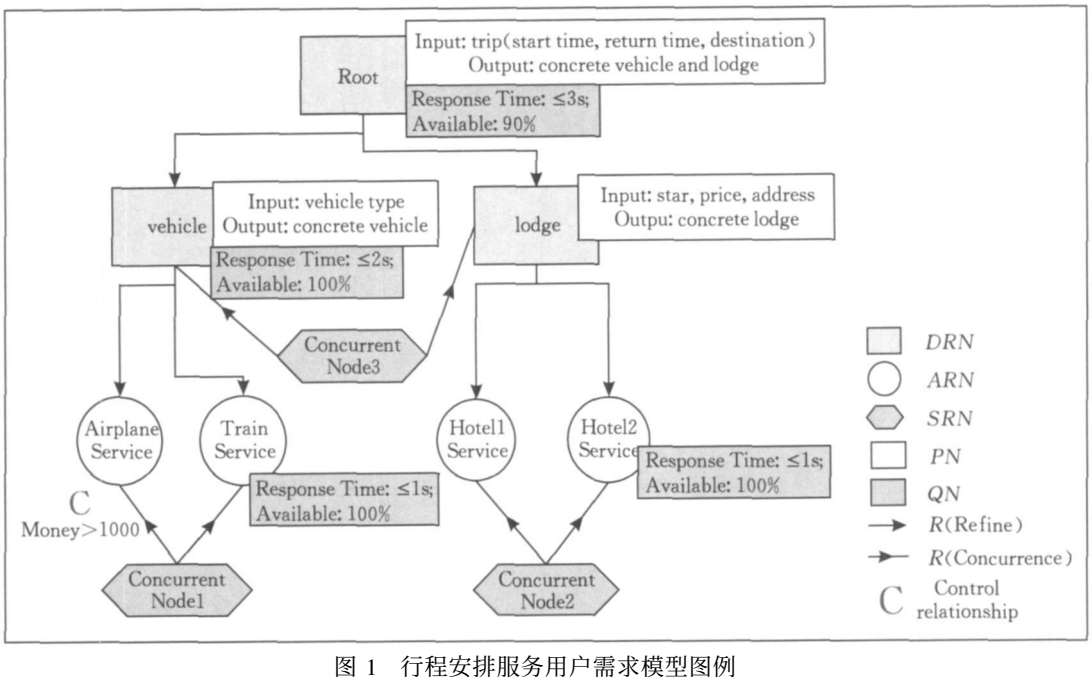


图 1 行程安排服务用户需求模型图例

如果不考虑图 1 中的六边形节点(*SRN*)和相应的 *R*(Concurrence)边，图 1 即为一个树形结构的用户需求模型，从目标节点(*DRN*)开始，用户需求依据功能逐层细化，目标中包含的约束条件也相应分解并向下层传递，细分成若干包含目标、*Input*、*Output* 等(*PN* 属性)和负载、性能、价格等(*QN* 属性)的子描述需求节点.这个划分可以一直进行下去，直到整棵树的叶子节点(*ARN*)能够与现实中存在的服务相匹配.同时引入的 *SRN* 和 *R* 边表达了节点之间的结构关系，为测试提供了大量的有用信息.

因此，通过逐层分解细化的方法所建立的用户需求树，在笼统、粗略的用户需求和实际的 Web 服务之间建立了对应关系，根节点为用户需求，叶节点为原子服务，而中间节点对应多个原子服务的组合过程，另外节点间的各种依赖关系和约束条件也在需求树中得以体现，从而为测试用例的生成及选择提供了输入输出逻辑关系、服务组合时的控制流数据流关系等至关重要的信息.另外，当用户需求发生变化时，通过逐层传递，一定会影响到所选择的服务，经过评估后，如果变化太大，则需要重新选择服务(此为服务匹配和发现的过程)[14-15]；如果变化在可以修改的范围内，则能够迅速定位到需要改动的流程、位置，并做出适当修改，再进行相应的回归测试.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

4　基于用户需求树的 Web 服务测试

4.1　测试用例生成选择

用户需求模型建立以后，以此作为组织、实施测试的依据，即对照每一项需求，进行测试用例的生成和选择，着重覆盖用户所关注的功能，能节约测试资源、提高测试效率. 从用户需求模型图可以看出，各层需求之间存在包含关系，子节点是父节点的子集，因此，依据“尽量选择能覆盖更多用户需求的测试用例”这一指导原则，进行测试用例约简时，首先须考虑根节点的需求. 如果设计的用例能满足要求，那测试的效率自然很高. 但由于根节点需求的表述比较笼统和宽泛，可能会有误差产生，造成精度上的损失，为此还需要分别测试相关的中间节点甚至叶节点.

(1) 根节点级测试用例生成

首先需要进行根节点需求的测试，即在用户所能接受的时间、价格等限制条件下，看用户的需求是否被满足. 测试用例形如(*InputSet*, *OutputSet*)，*InputSet* 为输入集，即根据需求确定的测试数据，*OutputSet* 为输出集，即用户期望的输出数据和相关的限制条件. 测试用例根据用户需求(表示为若干有限制条件的类型、取值范围等因素)来生成，方法就是一般的等价类、边界值、决策表等. 例如测试“天气预报”这一服务，输入通常就是日期和地名，输出就是相应的天气情况(温度、湿度、风力等)和返回时间、价格等限制条件. 运行测试用例后，根据参数覆盖率准则来判断测试是否可以终止.

理想情况下，针对根节点需求的测试会开展得比较顺利，因为不涉及到具体的细节，只是一个目标级的验证. 但由于在实际测试过程中，会存在类型或范围表达的不一致或者实际输出与预期输出部分不一致等问题，难以判断是服务中存在故障还是测试用例设计有问题，需要沿着用户需求特征树，深入到子节点中去，做进一步的分析和测试.

(2) Web 服务组合测试用例生成

根节点和叶节点之间的中间层会涉及到多个服务的组合，多服务组合在一起时，可能会产生接口冲突(参数名、类型、个数不一致)、流程冲突(时序逻辑冲突、数据竞争等)等问题，因而需要进行针对组合服务的测试. 现有的方法大多偏重于测试控制流程，采用模型检查的思想，通过 Petri 网或自动机等模型对服务组合进行建模，以验证其可达性，或借助已有的一些模型检查工具生成测试用例.

我们曾开发了一个面向 Web 服务组合的、较为系统的数据流自动测试工具 DFTT4CWS 原型[16]，可以将 Web 服务自动转化为适用于数据流测试的模型，并在此基础上通过静态分析检测程序中可能存在的数据流异常，针对各种数据流覆盖策略自动生成测试路径，并为每条路径提供变量约束信息以生成测试数据. DFTT4CWS 能够给出 All-C-Uses-Paths、All-P-Uses-Paths、All-DU-Paths、All-Uses-Paths、All-Defs-Paths 等各种覆盖策略下的统计信息(参见表 1). 如图 2 所示，All-DU-Paths 是最强的数据流测试覆盖策略，包含了其它所有的数据流测试覆盖策略.

表 1　数据流测试中的几种覆盖策略

名称	含义
All-C-Uses-Paths	如果在节点 i 处对变量 v 进行了赋值，则测试集合中至少存在一条路径从节点 i 到该定义值的每一个用于计算的变量使用节点
All-P-Uses-Paths	如果在节点 i 处对变量 v 进行了赋值，则测试集合中至少存在一条路径从节点 i 到该定义值的每一个用于判断的变量使用节点
All-DU-Paths	如果在节点 i 处对变量 v 进行了赋值，则 All-du paths 要求从节点 i 到该定义值的每一个使用节点的每一条路径必须被覆盖，即该覆盖集合为变量的定义点到该定义值的使用点之间所有的路径
All-Uses-Paths	如果在节点 i 处对变量 v 进行了赋值，则测试集合中至少存在一条路径从节点 i 到该定义值的每一个使用节点，即要求变量的每一个定义值的每一次使用都得到执行
All-Defs-Paths	如果变量 v 在一个节点 i 中被定义，则测试集合中至少存在一条路径经过节点 i 到变量 v 的某个使用节点，即该覆盖策略要求每个变量的每一个定义值至少使用一次

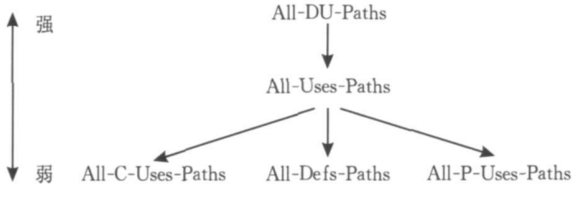


图 2　数据流测试覆盖策略关系

根据用户需求树，可以较容易地获知多个服务是如何组合的，中间节点和叶子节点的父子关系、叶子节点之间的兄弟关系非常明确，相关的属性、操作以及各种约束限制条件也能够被传递和继承. 因此，可着重针对这些属性、操作和限制，在用户需求树的驱动下，结合控制流数据流等信息进行测试路径和测试数据的生成选择：即根据用户需求树的结构建立控制流图，进行可达性分析，按照各种覆盖率的要求生成测试路径，并根据具体情况补充相应的测试数据；再根据用户需求树的数据依赖关系，运

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

用 DFTT4CWS 工具针对各种数据流覆盖策略自动生成测试路径，并为每条路径提供变量约束信息以生成测试数据.

如果运行测试用例，发现服务组合中有不可达的服务或组合冲突，需要对相应结构进行调整；或者发现现有服务不能满足需求，还需要删除冗余、错误的服务并补充新的合适的服务.

(3) 利用节点信息进行测试用例生成选择

组合服务测试更多侧重于多个服务之间的联系，单个服务的质量有时也需要进一步保障. 我们定义了 Web 服务的一种抽象模型 ASM[17]，包含了服务的结构和语义信息，形如 ASM={DT, WSN, OPSet {OPN, INPS {*PN*, *C*}, OUTPS {*PN*, *C*}}}，分别表示{服务文本描述，服务名称，服务操作集{操作名称，输入操作集{参数名，参数的本体概念}，输出操作集{参数名，参数的本体概念}}}. 同样，可用此模型来表示用户需求树上的叶子节点，据此来实现服务的发现和匹配，从而最终确定所选用的各单个服务.

在此基础上，进行针对单个服务的测试. 用例的生成和选择与现有的方法差别不大，也即首先根据服务的 WSDL 文档获知服务名称、输入输出参数的名称和类型，运用等价类划分、边界值、决策表等方式来生成初始的测试数据，再结合用户的需求(包括目标、*Input*、*Output*、前置后置条件、QoS 要求等)，通过变异测试[9-11]的方法来筛选用例，产生一些变异子，如参数个数的变异、参数次序的变异、参数类型的变异、参数范围的变异、参数约束条件的变异等，来检测测试用例的充分度，具体示例见表 2.

表 **2** 针对用户需求模型的变异操作(变异子)

名称	含义	示例
EP	Empty Parameter	airplane(Date depart, String address)变异为 airplane(null, String address) 或者 airplane("", String address)
EPO	Exchange Parameter Order	airplane(Date depart, String address)变异为 airplane(String address, Date depart)
IUO	Insert Unary Operator	pay(String cardNum, int amount)变异为 pay(String cardNum, int-amount)
PS	Parameter Shift	pay(String cardNum, int amount)变异为 pay(String cardNum, int amount ≫1)
PLT	Parameter Length Transfom	login(String name, String passwd)，若有约束 stringLength (?passwd, 6)，6变异为7
PA	Parameter Assignment	pay(String cardNum, int amount)，若有约束 lessThan(?amount, 2000)，变异为 pay(cardNum, 2000)

4.2 回归测试波动分析

回归测试是对修改后的软件进行重新测试，以保证修改后的软件仍然可提供正常功能，满足预期的需求. 为节省测试成本，回归测试只需要测试那些由于修改而受到影响的部分[18,19]. 波动分析是回归测试的基础，能够发现系统发生修改和受系统修改所产生的影响等情况，从而对此进行回归测试. 当用户需求发生变动时，对应的服务也要进行相应的修改，根据用户需求特征树可以迅速定位服务中的相关路径、元素和变量，即通过波动分析来寻找发生修改的服务以及和这些服务有依赖或关联关系的服务，以此作为回归测试的对象，从而保证全面、有效的回归测试.

用户需求特征树模型能反映需求-服务的关联关系，即根节点为需求，叶节点为服务，中间节点是纽带，相应地，可以据此分析需求变动和测试用例之间的关系，作为回归测试测试用例集约简的依据；同时度量相关服务受需求变动影响的程度，使受影响大的服务得到优先测试. 在用户需求特征树模型中，从发生变动的需求节点出发，沿着边可以找到一条或多条发散的路径到达叶节点，沿路径的所有服务(单个服务或者组合服务)都不同程度地受需求变动影响，称这些路径为需求变动影响路径，反映了需求变动影响服务的轨迹.

为形式化描述上述过程，先给出如下定义.

定义 **2**. 需求变动的基本动作类型有 4 种，形如 *ChangeRequest*={*Add*, *Delete*, *Combine*, *Decompose*}，分别表示增加、删除、合并和拆分需求. 在此基础上可以衍生出更为复杂的需求变化情况，例如某些需求的变化就可以视作先删除旧的需求再增加新的需求.

根据需求变动的基本类型，可以确定需求变动源点. 由分析可知：*Add*, *Delete*, *Decompose* 都分别对应某一个原子需求节点，而 *Combine* 则对应两个原子需求节点的公共父节点.

定义 **3**. 需求变动影响路径 *Path* 是用户需求特征树上的一条从根节点到叶节点的路径，建立了需求变动与对应服务的关联关系.

根据波动分析的理论，需要进行回归测试的服务除了直接被修改的服务以外，考虑到依赖关系的

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

传递性, 与被修改服务有依赖关系的服务以及与被修改服务有新产生(或新消灭)关系的服务也需要进行回归测试. 而这些依赖关系以及对应的需求变动影响路径可以通过遍历需求树来确定. 如算法 1 所示, 为自动检索出需求变动对相关服务的影响, 可以在用户需求模型中以变动点为源, 使用有向图的广度优先遍历算法查找出变动点影响到的原子需求服务对象. 根据用户需求变动的不同类别(*Add*, *Delete*, *Combine*, *Decompose*), 先分别定位到相关的变动源, 再根据上述算法查找出变动所直接影响到的原子需求服务, 进而依据用户需求模型进行波动分析, 即可获知间接影响到的其它服务以及各相关路径.

算法 **1**.　获取用户需求变动所影响到的原子服务集.

```
输入: G(用户需求模型), Node(需求变动源点)
输出: InfluenceSet(受变动影响的服务集, 初始为空)
for each v∈G
  set visited[v]=false;
  InitQueue(Q);
  visited[Node]=true;
  EnQueue(Q, Node);    //Node 入队列
  while(!QueueEmpty(Q)){//队列不空
    u=DeQueue(Q);      //取队列首元素
    for each m 邻接 u{
      if(!visited[m]){
        visited[m]=true;
        if m.NodeType==WS
          then InfluenceSet=InfluenceSet∪m;
          else EnQueue(Q, m);
        }
      }
    }
return InfluenceSet
```

假设 $Paths=\{Path_1, \cdots, Path_n\}$ 为关于某需求变动的所有影响路径的集合, 在此路径集合中统计除根节点以外各节点出现的频次, 频次越高的节点, 说明受影响程度越大, 需要复用更多的测试用例以加强测试. 因而可以依据受需求变动的影响度, 对用例排序, 从而实现回归测试用例集的优化.

5　实例分析

本节我们首先给出一个对 Web 服务进行测试的实例, 以说明我们实验的步骤以及环境配置; 进而结合图 1 中的行程安排实例, 按照第 4 节中的方法, 进行实例展示和分析.

5.1　测试流程示例

我们使用机票查询 Web 服务(http://webservice.webxml.com.cn/webservices/DomesticAirline.asmx)作为测试对象. 在本实验中, 机票查询 Web 服务需要 3 个输入参数: 出发城市、抵达城市和日期. 通过用户需求模型的分析, 我们发现: 出发城市和抵达城市需要在系统支持的城市名称集合中; 日期需要大于当前日期, 格式为 yyyy-MM-dd (如 2012-12-12). 另外, 返回结果中, 如果日期格式不正确, 则返回"日期格式错误", 如果日期小于当前日期, 则返回"日期错误"; 如果出发或者抵达城市不在城市名称集合中, 则返回"出发或抵达的城市不被支持", 如果查询出的航班为空, 则返回"没有航班"; 否则, 返回所有航班信息的列表.

实验中, 我们使用 SOAP 消息对测试用例进行封装, 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. 为了防止服务器负载过大, 两个 SOAP 消息之间会间隔一段时间(实验中设置为 1s), 处理流程包括测试用例的生成选择、测试执行和测试结果分析, 具体表述如下:

1. 从支持的城市名称集合中选择出发城市和抵达城市, 并生成符合格式等要求的日期;

2. 随机生成两串中文字符串, 作为出发城市和抵达城市; 随机生成一串字符串作为日期;

3. 生成测试用例, 输入数据从步 1、2 得到的数据中选取, 并生成预期的返回结果;

4. 新建一个线程, 使用 SOAP 对测试用例进行封装, 并发送到服务器端, 等待并解析服务器端返回的结果;

5. 将返回的结果和预期的结果进行比较, 判断返回结果的正确性, 如两者一致, 则说明返回结果正确, 否则, 说明返回结果错误;

6. 将线程阻塞 1s, 然后重复步 3~5, 直到达到预先设置的测试次数;

7. 将实验的结果写入相应的 xml 文件中.

我们一共对机票查询 Web 服务进行了 100 次测试, 其中, 正例(城市和日期正确)60 次, 反例 40 次(其中, 单出发城市错误的 10 次, 单抵达城市错误的 10 次, 出发和抵达城市同时错误的 10 次, 单日期错误的 10 次). 实验得到的结果如表 3、表 4 和表 5 所示.

表 **3**　参数正确实验结果表

实验信息	实验次数	返回航班信息	返回"没有航班"	返回结果错误
参数正确实验	60	19	40	1

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

表 4 城市参数错误实验结果表

实验信息	实验次数	返回“出发或抵达的城市不被支持”	返回结果错误
出发城市实验	10	10	0
抵达城市实验	10	10	0
出发和抵达城市实验	10	10	0

表 5 日期参数错误实验结果表

实验信息	实验次数	返回“日期格式错误”	返回“日期错误”	返回结果错误
日期格式实验	8	8	0	0
日期小于当前日期实验	2	0	2	0

表 3 显示的是在输入参数符合类型要求和潜在要求的情况下实验的结果信息，即正例测试. 在这 60 次实验中：返回航班信息的有 19 次；返回“没有航班”信息的有 40 次，这有可能的确是两个城市间或者当天没有航班，但也有可能是该 Web 服务有错误，属于待定状态，需要进一步分析；返回结果有误的有 1 次，该测试用例如图 3 所示，返回结果显示“日期格式错误”，不同于预期输出结果，显然是该服务存在问题. 表 4 和表 5 是通过反例测试所得到的结果，预期输出和实际输出都相吻合，没能检测出服务中的其它问题. 由此可见：实验中该机票查询 Web 服务的正确率为 59%，错误率为 1%，不确定率为 40%. 实验检测出该 Web 服务功能上的 1 项错误：当输入日期为闰日时，该服务不能返回正确结果.

```
〈TestCase〉
  〈startCity〉广州〈/startCity〉
  〈lastCity〉沈阳〈/lastCity〉
  〈theDate〉2012-02-29〈/theDate〉
  〈expect_Result〉航班列表信息〈/expect_Result〉
〈/TestCase〉
```

图 3 机票查询 Web 服务中的测试用例示例

实验表明：在确定了服务的输入输出参数类型、限制条件以及输入输出之间的部分逻辑关联关系以后，尚且有较高的不确定率（本次实验中为 40%）且错误发现率也比较低，说明测试的效率不高；如果只有输入输出参数的类型信息（现有方法大都如此，例如工具 soapUI 就是根据 WSDL 文档得到参数类型信息，然后再让用户随机填写测试数据），再据此生成测试用例用于测试，那测试的效率会更低，因为有太多的不确定情况，无法确定是服务的问题还是用例的问题.

5.2 用户需求驱动的行程安排服务测试示例

从图 1 可知，行程安排中的用户需求首先表示为根节点 Root，其 *PN* 属性包含了用户行程的出发时间、返回时间、目的地等信息，另外，根节点划分成 vehicle、lodge 两个分部，之后并发地进行两个流程：(1) 航班（车次）选择流程. 该流程根据 vehicle 选择调用 Train Service（TS）或是 Airlines Service（AS），被调用的服务将根据用户输入的 *PN* 寻找合适航班或车次；(2) 旅馆选择流程. 该流程调用已有的两家旅馆查询服务 Hotel1 Service（H1）和 Hotel2 Service（H2），根据 *PN* 和用户在食宿方面的要求选出合适的旅馆客房.

5.2.1 需求驱动的测试用例生成选择

根据第 4 节中提到的方法和步骤，首先根据图 1 中用户需求模型生成选择测试用例.

第一是进行根节点级的目标验证. 用户输入 *PN*、vehicle 和 lodge 的要求，该服务返回行程安排推荐列表或者无法返回结果，其中：*PN* 中至少有出发时间、返回时间、起始地、目的地这 4 项内容，取值范围广泛；vehicle 有 1 项内容，2 种取值（飞机或火车）；lodge 有 1 项内容，2 种取值（H1 或 H2）.

这个例子涉及 6 个输入参数，参数类型有日期型、字符串型、枚举型，通常的方法是：根据参数类型，运用等价类划分、边界值选取、决策表等策略来生成测试用例，可生成若干符合参数类型限制的测试用例. 如果用这些用例来测试，只符合了类型条件的限制，而没有考虑到服务内部的潜在限制条件，大多数用例将无法返回正确的结果（例如，起始地为“abcde”，符合字符串类型的要求，但提交给该服务时，输出一定是“无法返回结果”），这将难以判断是由于用例设计有误所导致的后果还是 Web 服务本身就存在问题.

而根据用户需求模型，我们除了可以获知参数类型外，还能够得到一些约束条件（有些属于常识范畴，有些是用户特别要求的条件），比如返回时间要晚于出发时间，起始地、目的地为地理上确有的城市名，等等. 直观上看，在生成用例时，如果增加了约束条件，自然会使得参数的取值范围显著缩小，运行同样数量的测试用例，我们的方法检测错误的效率更高.

针对行程安排实例，我们对等价类划分中随机选取测试数据的策略（Random）和运用用户需求模型生成测试用例的策略（UM）进行了比较，考虑不同数量的测试用例对根节点级服务中 4 个原子服务

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

(TS, AS, H1, H2)的覆盖情况.其中, Random 方法是在 *PN*、Vehicle、Lodge 所对应的等价类{合法 *PN*, 非法 *PN*}、{非法 Vehicle, AS, TS}、{非法 Lodge, H1, H2}中随机选择测试数据, 以得到若干不同数目的测试用例集; 而 UM 方法能保证输入信息的合法性和合理性, 即时间格式、地点名称等都符合服务的内在要求, 只需要在等价类{合法 *PN*}、{AS, TS}、{H1, H2}中随机选择测试数据, 并补充一些异常情况做为反例来考查服务在错误输入的情况下返回结果的情况, 另外还可以考虑 Vehicle 和 Lodge 这两个子服务并发执行的情况, 即输入分别为(*PN*, Vehicle)、(*PN*, Lodge)以及(*PN*, Vehicle, Lodge)的情况.具体测试过程是: 分别进行 10 组实验, 各组采用的测试用例数目分别为 2、4、6、8、10、12、14、16、18、20, 得到各用例数目下对各原子服务的执行覆盖情况(其中, 覆盖率分别为 25%、50%、75%、100%, 表示相应执行了 1 个、2 个、3 个和 4 个原子服务); 为减少实验的偶然性, 再重复这 10 组实验 8 次, 取其平均值, 作为一般情况下根节点服务中原子服务被执行到的情况, 具体结果如图 4 所示: 随着测试用例数目的增加, 对 4 个原子服务的执行覆盖率也在逐步增加, 用 UM 方法, 平均执行 14 条用例就能全部执行根节点服务中的原子服务; 而用 Random 方法, 需要平均执行 18 条用例, 才能达到对原子服务的 100%覆盖.另外, 图中各测试用例数目下的覆盖率均表明 UM 方法要优于 Random 方法, 在同样数量的用例下, UM 方法能达到更高的覆盖率.

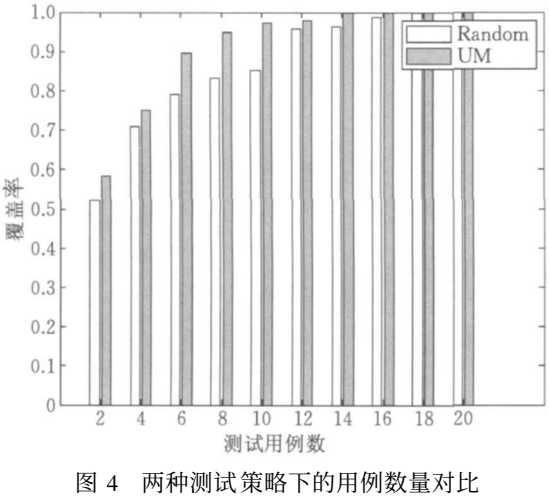
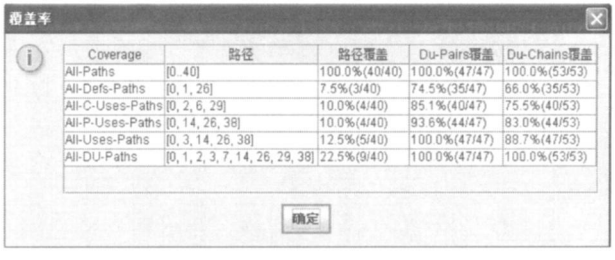


图 4　两种测试策略下的用例数量对比

第二是进行中间节点级服务组合测试.从图 1 获知节点之间的控制依赖关系为{并行结构: (Train ‖ Airplane) ‖ (Hotel1 ‖ Hotel2), 约束条件: if 票价预算 Money> 1000 元 then 选择 Airplane; else 选择 Train}, 数据依赖关系为{root => vehicle => Train, root => vehicle => Airplane, root => lodge => Hotel1, root => lodge => Hotel2}.据此进行静态分析, 得到 CFG 图和定义-使用链, 进而可以得到满足各覆盖率要求的相关测试路径, 再补充测试数据, 即可得到相应的测试用例.例如, 使用我们的工具 DFTT4CWS 得到的在不同数据流覆盖策略下的统计结果如图 5 所示, 该服务组合的业务流程共有 40 条可执行路径、47 对有效定义-使用对(Du-Pairs)和 53 条有效定义-使用链(Du-Pairs).为便于表示, 图 5 中的路径是实际路径的代号.只需 9 条路径(所有流程执行路径的 22.5%)即可覆盖该服务组合流程中的所有有效定义-使用链, 该集合即为强度最高的数据流覆盖策略 All-DU-Paths, 达到对数据使用的测试要求.


覆盖率

Coverage	路径	路径覆盖	Du-Pairs覆盖	Du-Chains覆盖
All-Paths	[0, 40]	100.0%(40/40)	100.0%(47/47)	100.0%(53/53)
All-Defs-Paths	[0, 1, 26]	7.5%(3/40)	74.5%(35/47)	66.0%(35/53)
All-C-Uses-Paths	[0, 2, 6, 29]	10.0%(4/40)	85.1%(40/47)	75.5%(40/53)
All-P-Uses-Paths	[0, 14, 26, 38]	10.0%(4/40)	93.6%(44/47)	83.0%(44/53)
All-Uses-Paths	[0, 3, 14, 26, 38]	12.5%(5/40)	100.0%(47/47)	88.7%(47/53)
All-DU-Paths	[0, 1, 2, 3, 7, 14, 26, 29, 38]	22.5%(9/40)	100.0%(47/47)	100.0%(53/53)

确定

图 5　各种数据流覆盖策略的统计结果

在实际运行测试用例后, 可能会产生一些问题, 例如, 假设 Money 为 1200 元, 那只会运行 Airplane 服务, 但可能当天机票已全部售出, 无法返回用户需要的航班预订信息, 此时还需要用户干涉, 再次选择 Train 服务.因此, 为提高用户的满意度, 需要进一步完善服务组合的流程, 当某个服务不能完成某任务时, 要能够自动启用备用服务或功能等价服务, 从而给用户提供尽可能多的选择.

第三是进行叶节点级测试用例的生成选择.以 Hotel1 服务为例, 输入为{出发时间、返回时间、目的地、星级、价格、地址}, 输出为{旅店名称、地点、到达日期、离开日期、单价、总价}, 根据变量类型和取值约束生成初始用例集, 运用等价类划分、边界值、决策表、整数线性规划等方式来生成初始的测试数据, 规模会非常庞大; 再根据表 2 中列出的变异子, 对相应参数的类型、数目、约束条件等做变异, 得到新的用例, 再根据其运行结果计算变异充分度, 从而筛选出数目合适的用例.其中的相关数据类型和约束条件如表 6 所示.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

表 6 Hotel1 服务中数据类型和约束条件示例

数据类型	参数示例	约束条件
Date	出发时间、返回时间、到达日期、离开日期	出发时间< 返回时间; 到达日期< 离开日期; 出发时间< 到达日期; 离开日期< 返回时间;
String	目的地、地址、旅店名称、地点	目的地瓤地址瓤地点(瓤表示行政区域上的包含, 而不是字符串意义上的包含); 旅店名称需要是客观存在的, 且和地点有一一对应关系;
Enum	星级	{0, 1, 2, 3, 4, 5}, 和旅店名称有一一对应关系;
Real	价格、单价、总价	价格≥单价; 总价= 单价×(离开日期-到达日期)(为简单起见, 暂不考虑 12 点后退房要多算半天的行规);

5.2.2 需求驱动的回归测试

当用户需求发生变动时, 势必会影响到某些服务, 需要进行回归测试, 为此要在原有测试用例集的基础上进行选择并适当新增用例. 例如: 在行程安排实例中, 用户已通过该服务组合完成了交通工具预订和旅馆预订, 但突然有需要提前返回, 即对需求中的"返回时间"输入项做了改变, 由此可能导致以下变化, 所涉及到的服务以及需要变化影响路径如下:

(1) 影响到飞机票或火车票的预订. 由于日期提前, 有可能导致订不到飞机和火车票, 需要增加交通方式(变动类型为 Add), 比如汽车 Bus. 因此需要增加服务 Bus, 相应的影响路径为 root-> vehicle -> Bus;

(2) 住宿天数的变化; 如果对住宿情况不满意, 也可能更换旅馆. 可能会会修改住宿天数或住宿地址(变动类型为先 Add 后 Delete). 即需要修改服务 Hotel1 或 Hotel2, 相应的影响路径为 root -> Lodge-> Hotel1(或 Hotel2).

(3) 对业务流程的修改. 原先航班(车次)选择流程和旅馆选择流程是并发执行的, 但由于返回日期修改可能会导致订不到票, 相应会改变住宿计划, 所以这个流程要由并行改为串行, 先订票, 再订旅馆, 这样更为合适. 即需要修改服务流程(变动类型为先 Combine 后 Decompose), 更改为 root -> vehicle-> Lodge.

例如, 如果行程安排服务的流程发生了变化(从并行变成了串行), 重点是找出变化所带来的影响, 具体的测试执行过程和 5.1 节中的流程类似. 相应的回归测试过程如下:

1. 执行 root-> vehicle 的测试, 具体过程如 5.1 节中的流程所示, 但要注意日期的范围限制, 且测试次数设为 1;

2. 视 vehicle 的输出情况来决定是否运行 Lodge, 即: 若 vehicle 输出为航班或车次信息, 则转步 3; 若 vehicle 输出为空或出错信息, 则转 4;

3. 执行 vehicle-> Lodge 的测试, 具体过程也类似于 5.1 节中的流程, 只是在输入输出参数上有所变化, 且测试次数设为 1;

4. 合并两次测试的结果, 即: 若 Lodge 被执行, 将步 1 和步 3 中的结果合并后输出; 若 Lodge 没被执行, 直接输出出错信息.

根据上述过程, 我们在 Intel Core2 Q8400 CPU、4G RAM(系统实际占用 3.58G)、Windows 7 32bit 操作系统、JRE 6.0 Update 23 32bit、axis 1.4 的运行环境下分别运行了 60 条正例和 40 条反例(分属于 5 组测试用例集, 对应于表 7 中的 5 行), 得到串行处理的消耗时间(不考虑线程阻塞的时间, 由于单个用例的执行时间很短, 故记录运行每组测试用例集所消耗的总体时间, 包括生成、发送、接受 SOAP 消息并处理返回结果的时间), 并统计导致 Lodge 未被执行的测试用例数目. 为了便于比较, 我们还计算了并行处理的消耗时间(同样, 也不考虑线程阻塞的时间, 消耗时间的计算方法也和串行处理的一致), 具体结果如表 7 所示.

表 7 并行、串行方法消耗时间结果表

实验信息	用例数目	并行处理消耗时间/ms	串行处理消耗时间/ms	导致 Lodge 未被执行的用例数目
参数正确实验	60	1260	1079	43
出发城市实验	10	232	133	10
抵达城市实验	10	241	134	10
出发、抵达实验	10	236	142	10
日期实验	10	202	112	10

根据表 7 中的数据, 我们做如下分析:

(1) 在同样的实验次数条件下, 实验消耗时间略有浮动, 这主要是由于网络访问时间不确定造成的.

(2) 实验结果表明, 串行处理方法消耗时间比并行处理方法少很多, 与通常情况下并行处理效率更高的认知大相径庭. 结合该实例, 我们分析原因是: 串行方法虽然增加了同步和切换开销, 但由于增加了限制条件"若前面服务的返回结果不合要求, 则不执行后面的服务", 使得部分正例和全部的反例都不用执行对 Lodge 的测试, 从而导致其花费时间小于并行处理方法所消耗的时间.

(3) 冗余测试用例(输入不同, 输出相同)在测

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

试用例集中占有很大的比例(本例中，71.67%的正例和 100%的负例均为冗余测试用例，即输出为空或出错信息).在某些情况下，通过增加一些限制条件(本例是并行改串行)，可以约简冗余，避免测试资源的浪费.

6　结束语

Web 服务涉及到开发者、提供者、集成者、第三方确认者和用户等多方，通常只提供接口信息给用户，一般进行黑盒测试.为提供功能更复杂的服务，需要多个服务的组合，但由于不能充分信任各原子服务并且多个服务组合时会带来流程冲突、数据冲突等问题，目前 Web 服务测试在模型表示、测试方法等方面还需要进一步的研究.

在现有工作的基础上，通过用户需求来驱动 Web 测试工作，以更好地满足终端用户的需求并提升测试工作的效率.即首先建立了用户需求特征模型，分层次、多维度地刻画用户需求，将笼统抽象的目标需求逐渐分解精化为具体明确的原子 Web 服务，层次关系、控制结构、约束条件的引入使得模型能真正反映用户的需求并有一定的可扩展性，从而可以很好地将需求和实现联系起来；其次，根据用户需求模型组织、实施测试，分别对照目标级需求、服务组合需求以及原子服务需求，进行测试用例的生成和选择，即只覆盖用户关注的语句或路径，有效提高测试效率，既能覆盖需求又能覆盖实现(服务组合、路径、数据流等)；另外，利用用户需求模型中的依赖关系，在用户需求发生变动时，能迅速定位到服务的相关路径、元素和变量，从而有利于回归测试的用例选择和优化；最终，通过行程安排实例系统地展现我们在模型表示、测试用例生成选择、回归测试等方面的工作，表明用户需求驱动的 Web 服务测试有良好的实施效果.

在后期工作中，将进一步完善我们的原型工具，在图形界面表示、算法的通用性研究、实例的规模和体现度等方面都力争更进一步.

参　考　文　献

[1] Yu W D, Supthaweesuk P, Aravind D. Trustworthy Web services based on testing//Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering (SOSE' 05). Shanghai, China, 2005: 159-169

[2] Canfora G, Di Penta M. Testing services and service-centric systems: Challenges and opportunities. IT Professional, 2006, 8(2): 10-17

[3] Yang Li-Li, Li Bi-Xin. Testing Web services: A review. Computer Science, 2008, 35(9): 258-265(in Chinese)
(杨利利，李必信. Web 服务测试问题综述. 计算机科学，2008，35(9)：258-265)

[4] Li Zhong-Jie, Sun Wei, Jiang Zhong-Bo, Zhang Xin. BPEL4WS unit testing: Framework and implementation//Proceedings of the 2005 IEEE International Conference on Web Services (ICWS' 05). Orlando, Florida, USA, 2005: 103-110

[5] Wang Yong-Bo, Bai Xiao-Ying, Li Juan-Zi, Huang Ruo-Bo. Ontology-based test case generation for testing Web services//Proceedings of the 8th International Symposium on Autonomous Decentralized Systems (ISADS' 07). Sedona, Arizona, USA, 2007: 43-50

[6] Ruth Michael, Tu Sheng-Ru. Concurrency issues in automating RTS for Web services//Proceedings of the 2007 IEEE International Conference on Web Services (ICWS' 07). Salt Lake City, UT, USA, 2007: 1142-1143

[7] Liao Jun, Tan Hao, Liu Jin-De. Describing and verifying Web service using Pi-calculus. Chinese Journal of Computers, 2005, 28(4): 635-643(in Chinese)
(廖军，谭浩，刘锦德. 基于 Pi-演算的 Web 服务组合的描述和验证. 计算机学报，2005，28(4)：635-643)

[8] Tsai W T, Chen Y, Paul R. Specification-based verification and validation of Web services and service-oriented operating systems//Proceedings of the 10th IEEE International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS 2005). Sedona, Arizona, USA, 2005: 139-147

[9] Jiang Yin, Xin Guo-Mao, Shan Jing-Hui, Zhang Lu, Xie Bing, Yang Fu-Qing. A method of automated test data generation for Web service. Chinese Journal of Computers, 2005, 28(4): 568-577(in Chinese)
(姜瑛，辛国茂，单锦辉，张路，谢冰，杨芙清. 一种 Web 服务的测试数据自动生成方法. 计算机学报，2005，28(4)：568-577)

[10] Wang Rui, Huang Ning. Requirement model-based mutation testing for Web service//Proceedings of the 4th International Conference on Next Generation Web Services Practices (NWES' s 08). Seoul, Korea, 2008: 71-76

[11] Lee S F, Bai X Y, Chen Y N. Automatic mutation testing and simulation on OWL-S specified Web service//Proceedings of the 41st Annual Simulation Symposium (ANSS-41 2008). Ottawa, Canada, 2008: 149-156

[12] Zhang Xiao-Fang, Chen Lin, Xu Bao-Wen, Nie Chang-Hai. Survey of test suite reduction problem. Journal of Frontiers of Computer Science & Technology, 2008, 2(3): 235-247(in Chinese)
(章晓芳，陈林，徐宝文，聂长海. 测试用例集约简问题研究及其进展. 计算机科学与探索，2008，2(3)：235-247)

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

[13] Gu Jing-Xian, Xu Lei, Xu Bao-Wen. Coverage criteria and test requirement reduction for component-based Web application. Journal of Southeast University (English Edition), 2010, 26(1): 36-42

[14] Plebani Pierluigi, Pernici Barbara. URBE: Web service retrieval based on similarity evaluation. IEEE Transactions on Knowledge and Data Engineering, 2009, 21(11): 1629-1642

[15] Feng Zai-Wen, He Ke-Qing, Li Bing, Gong Ping, He Yang-Fang, Liu Wei. A method for semantic Web service discovery based on context inference. Chinese Journal of Computers, 2008, 31(8): 1354-1363(in Chinese)
(冯在文, 何克清, 李兵, 龚平, 何扬帆, 刘玮. 一种基于情境推理的语义 Web 服务发现方法. 计算机学报, 2008, 31(8): 1354-1363)

[16] Hou Jun, Xu Lei. DFTT4CWS: A testing tool for composite Web services based on data-flow//Proceedings of the 2009 6th Web Information Systems and Applications Conference (WISA 2009). Xuzhou, Jiangsu, China, 2009: 62-67

[17] Chen Lianjie, Xu Lei. A framework for Web service discovery based on ontology similarity//Proceedings of the 5th IEEE International Symposium on Service-Oriented System Engineering (SOSE 2010). Nanjing, Jiangsu, China, 2010: 197-201

[18] Li Zheng, Harman Mark, Hierons Robert M. Search algorithms for regression test case prioritization. IEEE Transactions on Software Engineering, 2007, 33(4): 225-237

[19] Xie Kai. Component system regression testing model and technology research[M.S. dissertation]. Southeast University, Nanjing, 2006(in Chinese)
(解凯. 构件系统回归测试模型与技术研究[硕士学位论文]. 东南大学, 南京, 2006)

XU Lei, born in 1978, Ph. D., associate professor. Her current research interests include Web service analysis and testing.

CHEN Lin, born in 1979, Ph. D.. His current research interests include software analysis and software refactoring.

XU Bao-Wen, born in 1961, Ph. D., professor, Ph. D. supervisor. His current research interests include programming language, software engineering (software methodology, software analysis, software metrics and software testing), and Web technology.

Background

Web Services Testing is critical to ensure the qualities of Web Services and increase the trust of users. At present, researches are focused on test case generation and automatic testing. However, since Web Services only provide interface information to users, such as WSDL documents, BPEL, and OWL-S documents, the inner information of each Service is hard to obtain by testers, except the testers also are developers. So the testing methods are usually in the way of black-box. And we wish to testing Web Services in the way of grey-box by increase the constrain conditions, which are drawn from the user requirement model. This paper presents the testing process for Web Services, which is driven by the users' requirements.

The work in this paper is partially supported by the National Natural Science Foundation of China (60873050, 90818027, 61003020, 91018005), the National Basic Research Program (973 Program) of China under grant No. 2009CB320703, and the Opening Foundation of State Key Laboratory of Software Engineering in Wuhan University (SKLSE20080717).

These projects focus on Service Quality, Web Service discovery and matching, testing for server side programs, testing driven by requirements, regression testing, and etc. And the group has done some work in the area of software engineering, such as Web application testing, Web service discovery and matching based on semantic information, data race detection for BPEL flows, and automatically generating testing paths for Web Service combination based on data flow. This paper aims to Web Service testing driven by user requirement, which including test case generation and selection in different requirement levels. Regression testing also can be carried out more efficiently based on the methods.

?1994-2014 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net