Web服务测试问题综述*)

杨利利　李必信
(东南大学计算机科学与工程学院　南京 211189)

摘　要　近来出现了一种新的支持分布式计算的范型——面向服务的体系结构(SOA)。Web服务就是这种结构的具体实现形式之一。众所周知,为了获得一个可信的、可靠的服务,对服务实施完全、充分的测试是至关重要的。所以本文对Web服务的测试方法和技术进行了调查研究。本文从SOA体系结构下Web服务的特点及其测试的新的挑战出发,讨论了Web服务测试与传统测试的不同点;接着从多个不同的角度(测试的视角和测试的策略)讨论了Web服务测试的相关问题。然后给出了一个Web服务测试过程的组织框架,还讨论了目前Web服务测试的研究现状和一些代表性的Web服务测试技术。最后总结全文并给出未来的研究方向。

关键词　面向服务的体系结构(SOA),Web服务,测试

Testing Web Services: A Review

YANG Li-li　LI Bi-xin
(School of Computer Science and Engineering, Southeast University, Nanjing 210096, China)

Abstract　Service-oriented Architecture (SOA) has recently emerged as a new promising paradigm for supporting distributed computing. Web services are special substitutes for this kind of architecture. Then as we all know, it is crucial to test Web services entirely and thoroughly to ensure trustworthiness and reliability. So we make a survey of some methods and technologies for testing Web services. This paper talks about the characteristics of Web services based on SOA and new challenges posed for testing firstly, then it compares Web services testing to traditional software testing. It also discusses the main problems of testing Web services from different views (testing perspectives and testing strategies), next introduces a model for organizing and managing Web services testing process. Also the related research on testing Web services and some representative testing technologies are pointed. At last, it gives conclution and discusses future research and practice on testing Web services.

Keywords　Service-oriented Architecture (SOA), Web services, Testing

1　引言

分布式计算由于在Internet应用方面的特殊优势,长期以来一直受到人们的普遍重视。然而,诸如CORBA,RMI和DCOM等传统的分布式计算模型都存在许多不足,它们要么过于复杂、容易出错、无法穿越防火墙正常工作,要么就是无法升级以支持大型的分布式网络上的应用。而且,它们在交互性方面对Internet的支持也较弱,越来越不能满足Internet上结构复杂、与平台无关的分布式计算要求。Web服务的兴起提出了面向服务的软件体系结构(SOA),为Internet上的分布式计算提供了一种基于标准的、松散耦合的跨平台的新范型。随着Internet应用的进一步发展,SOA这一新型的分布式计算模型必将得到广泛的应用。

然而,由于Web服务本身特殊的性质,很多传统的软件测试技术对于Web服务已失去其原有的功效。因此,我们需要研究针对Web服务的新的测试方法和技术,从而为服务软件的功能、性能、质量和可靠性方面提供有力的支持。

目前,国内外已经开始对Web服务测试进行研究并取得了一些初步的研究成果,本文的目的是对现有的一些Web服务测试方法和技术进行深入的剖析与探讨。首先讨论了Web服务和SOA体系结构的新特性及其测试的相关挑战,并比较了传统软件测试与Web服务测试的不同点;接着针对Web服务独特的性质,从多个不同的角度详细讨论了各种相关的测试问题和可能的解决方案。然后介绍了一种Web服务测试过程的组织框架,讨论了国内外的相关研究现状和典型的Web服务测试技术;最后总结全文并给出未来进一步研究的方向。

2　基本概念

2.1　Web服务和面向服务的体系结构

Web服务背后的概念非常复杂,不同的人对其持有的意见也不尽相同,因此对它的定义也有很多种。一个Web服务可看作是被URI识别的应用软件,其接口和绑定由XML描述和发现,并可与其它基于XML消息的应用程序交互[1]。Web服务也可看作是一种新型的Web应用程序,具有自包含、自描述以及模块化的特点,可以通过Web发布、查找和调用[2]。Web服务实现的功能可以是响应客户一个简单的请求,也可以是完成一个复杂的商业流程。一个Web服务配置好后,其它应用程序和Web服务就可以直接发现和调用该服

*)本文研究受到国家自然科学基金(60473065),江苏省自然科学基金(BK2007513)资助。**杨利利**　硕士研究生,研究方向是软件分析与测试;**李必信**　博士,教授,CCF高级会员,研究方向是软件分析与测试、程序切片技术和经验软件工程等。

务。具体来讲 Web 服务应具有如下特性:1)可描述。可以通过一种服务描述语言来描述;2)可发布。可以在注册中心注册其服务描述信息并发布;3)可查找。通过向注册服务器发送查询请求找到满足需求的服务,获取服务的绑定信息;4)可绑定。通过服务的描述信息生成可调用的服务实例或服务代理;5)可调用。使用服务描述信息中的绑定细节可以实现服务的远程调用;6)可组合。可以与其它服务组合在一起形成新的服务。

Web 服务采用了面向服务的体系结构(SOA)[3](见图 1)。

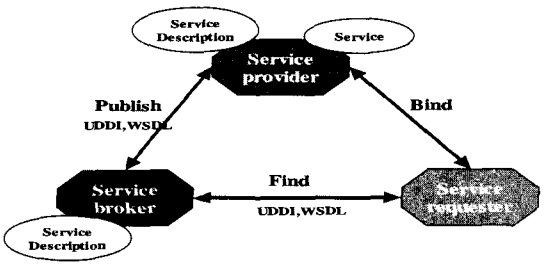

图 1 面向服务的体系结构(SOA)

该架构包含了三个实体和三个基本操作。这三个实体是:服务提供者(Service provider)、服务请求者(Service requester)和服务代理(Service broker)(也可称服务注册中心);三个基本操作分别是发布(publish)、查找(find)和绑定(bind)。

- 服务提供者是服务的所有者,通过向服务注册中心注册服务描述来发布服务,并通过服务访问平台提供服务;
- 服务请求者是需要特定功能的企业或组织,当服务请求者需要调用服务时,它首先利用服务代理提供的目录去搜索该服务,得到如何调用该服务的信息,然后根据这些信息去调用服务提供者发布的服务;
- 服务代理是存储服务描述信息的信息库,服务提供者在此发布其服务,服务请求方在此查找服务,获取服务的绑定信息。

SOA 体系结构中使用一系列标准和协议实现以上的功能,如:使用 WSDL(Web service description language)来描述服务,使用 UDDI(Universal description, discovery, integration)来发布和查找服务,而 SOAP(Simple object access protocol)则用于服务通信[4]。

2.2 Web 服务的特点及测试的主要挑战

与传统的组件相比,Web 服务有许多独特的地方,其中对测试有影响的方面主要体现在:

(1)SOA 结构下,服务请求者需要选择和调用 Web 服务,在使用中涉及 UDDI,SOAP,WSDL 等标准协议,但这些协议尚未成为成熟的产业标准。协议本身的验证也将是服务测试的一项内容。

(2)Web 服务的应用通常涉及到服务提供者、服务请求者和服务代理三种角色,他们应共同参与到测试的过程中。同时,由于他们间的分布式协作的特性使得测试的组织和管理变得更加困难。

(3)Web 服务的开发环境与实际运行环境有很大的不同。在服务发布之前,很难预测其实际的运行场景,如用户的类型、并发访问的数量以及用户使用环境(硬件设施、网络连接、操作系统、浏览器等等)的不同,这些差异和不确定性增加了 Web 服务测试的难度。

(4)分布式测试环境的多样性。不同的测试目标需要不同的测试床(测试环境),不同的测试阶段需要集成不同的系统。因此,必须在测试前仔细地规划好测试环境,并明确定义系统之间的相互依赖关系。

(5)SOA 体系结构下,Web 服务的发布、查找和绑定都是动态完成的,其过程的不确定性同样增加了服务测试的难度。

(6)Web 服务是包含大量运行时行为的分布式应用,大量的用户可能在同一时间访问同一服务,因此性能和可扩展性也是服务测试的重要方面。

(7)服务提供者发布服务的访问接口以便使用者调用,这一做法增加了服务的安全隐患,加大了系统被攻击的机会。因此,对服务进行测试还需考虑安全性问题。

(8)由于知识产权保护的问题,服务提供者不愿意暴露过多的服务实现的细节,服务以接口的形式发布,使用者通常只能获得 Web 服务的接口信息(WSDL 文档),因此对于除服务开发者外的测试者来说,一切白盒测试技术都将失效。

(9)Web 服务没有用户接口,因此对 Web 服务进行测试时需要构建测试客户端,由测试客户端本地或远程地调用 Web 服务,收集服务响应信息并分析测试结果。

(10)对用户而言,Web 服务是被调用的而非被拥有的,其演化发展的情况超出了用户的可控制范围,用户不了解服务何时发生演变(如版本升级或实现细节变更),因此服务的回归测试变得难以控制。

2.3 传统软件测试与 Web 服务测试

从 2.2 节讨论的 Web 服务测试的挑战可以看出,Web 服务测试与传统软件测试将存在很大的不同。表 1 给出两者之间的比较结果。从中可以看出,Web 服务测试需要服务提供者、服务中介和服务请求者协作共同完成测试任务,测试也由传统的、集中的、离线的测试变为分散的、在线的即时测试。

表 1 传统软件测试与 Web 服务测试

比较项	传统软件测试	Web 服务测试
测试参与者	独立的软件测试小组和软件开发人员共同负责软件的测试工作	需要服务提供者、服务中介和服务请求者共同参与到测试过程中
测试分布	集中的、多阶段的测试	分布的、远程的、多阶段的测试
测试模型	拥有软件的代码,可建立丰富的测试模型	不拥有服务的代码(服务开发者除外),有限的测试模型(可控制性和可观测性低)
测试覆盖	结构(白盒)和功能覆盖(黑盒)	除服务开发者拥有传统的覆盖范围外,服务提供者、服务集成者、请求者和服务中介可能只有黑盒覆盖范围
测试执行	离线的测试	在线的即时测试
测试客户端	可能是软件本身	需构建测试客户端调用被测服务
测试预言	软件的行为是可预知的,因此测试预言的生成较为容易	由于 Web 服务动态绑定特性,很难预测其实际的运行情况,测试预言的生成比较困难
回归测试	离线的回归测试;了解软件变更情况,能及时进行回归测试	在线的、基于运行时收集数据的测试;除服务开发者、服务提供者外,服务请求者、服务集成者和服务中介并不掌握服务的演化情况,回归测试需要额外的信息支持

3 Web 服务测试问题讨论

3.1 测试的视角

(1)服务开发者

作为服务的开发者,他们主要的目标就是要发布一个高可靠性的服务。因此,在开发过程中,他们测试服务,最大程度地发现程序中存在的缺陷。从这一点来讲,Web服务开发者与传统的软件开发者没有较大的不同点,他们同样都可以利用白盒测试技术检测程序中存在的错误;在程序做出修改后,为防止其引入新的错误,回归测试也是必需的。另外,服务开发者也可能试图评估服务的非功能属性及其处理异常情况的能力。尽管在这种情况下,测试代价比较有限(因为开发者测试自己的服务并不存在支付费用的问题),但是这样的非功能性测试却不能反映服务真正运行时的情况,因为它并没有考虑到提供者和使用者的基础设施配置、网络配置情况和负载等问题。

(2)服务提供者

服务提供者和服务请求者之间应就Web服务所保证的服务级别问题做出协商,这就是SLA合约。服务提供者应测试服务,以确保其符合SLA合约中服务级别的约定。但是与开发者不同的是,提供者只能采用黑盒测试的方法,因为服务对外只显示其接口信息。与服务开发者类似,提供者所进行的非功能性测试也未反映使用者的基础设施配置、网络配置情况和负载等真实的运行时情况。服务提供者了解服务升级和变更的情况,但不了解服务具体做了哪些修改,所以他们在服务变更后能及时地进行回归测试。

(3)服务集成者

服务集成者在构建自己的组合应用过程中进行测试的主要目的是为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。然而,SOA结构的动态绑定的特性给他们的测试工作带来了很大的挑战:1)被绑定的服务可以是众多候选服务中的任何一种,甚至是未知的服务;2)QoS测试需考虑所有可能的绑定;3)集成者并不知道服务是否已发生了变更,因此回归测试变得十分困难。

(4)UDDI注册中心

作为服务的发布者,UDDI注册中心需要对注册服务的质量进行评估,其评估结果将作为服务检索的依据。一般来说,主要从两个方面来考虑UDDI端的测试:一方面UDDI服务器需要保证被注册的服务具有一定的质量,另一方面UDDI服务器除了知道服务的具体供应商以外,还需要向服务请求者提供最能满足用户需求的服务。

(5)用户

就服务的使用者而言,他们仅仅关注在自己的应用中使用服务并能保证其能正常工作。对他们来说,SOA典型的动态特征既有其优越的地方,也有其令人不适的缺陷:一方面,动态性给应用带来了良好的性能、附加的特色或是开销的削减;另一方面,我们很难预测请求的响应时间和服务的可用性。此外,用户测试需要耗费大量的资源和花费:如用户在测试时需反复地调用服务,而有些服务对每次的使用都要收取一定的费用,这在一定程度上增加了用户端测试的开销。一种可能的策略就是服务提供者为用户测试引入特殊的考虑,如提供服务的部分功能供用户测试使用,或者限制用户的使用次数(当超过使用次数限制时才收取一定的费用)。

3.2 测试的策略

(1)易测试性

软件的易测试性是指使用一定的测试策略迫使软件中存在的故障被暴露的概率[5]。软件的易测试性设计是在软件的设计和编码中考虑测试问题,一般包括合约式设计(Design by Contract)、内建式测试和内建式自测试等几种方法。为了对Web服务进行充分、全面的测试,其易测试性问题也将是测试者重点考虑的问题。由于当前的Web服务描述文件(WSDL)中只包括了输入/输出的数量、每个输入/输出的变量类型、输入的顺序和输出返回的顺序以及如何调用该Web服务的信息,为了执行Web服务的黑盒测试和回归测试,以上这些信息还远远不够。现在已有一些学者正对如何扩展服务的描述文件进行研究,也有些研究人员提出由服务开发者提供一些测试用例,作为提供者/请求者间合约的一部分,并连同服务描述文档一同发布,以供集成者和用户测试使用。

此外,Web服务测试与传统软件测试的一个很大的不同在于Web服务实现时并未包含用户接口。因此,我们在测试Web服务前需要构建一个测试客户端,用于从测试人员那里获得测试输入,然后向服务发送请求并接收和解析服务的响应。

(2)功能测试

功能测试是Web服务测试中最基本的一项测试。如果某个Web服务不能正确地工作,那么它的性能、可靠性、安全和互操作性等问题将无从谈起。Web服务的功能测试的目标很明显,那就是保证Web服务对特定的用户请求能做出正确的响应。其测试的主要内容包括:对于允许的输入数据是否返回期待的输出?在边界值输入的条件下服务做出何种反应?当一个出错条件发生时将会出现什么样的状况?是否保证其预期的安全或授权要求?是否支持所有期望的通讯协议?

在这个过程中,作为服务的开发者,他们拥有服务的代码,可以为服务设计面向结构和面向功能的测试用例,在服务发布之前对其进行充分的测试以保证期望的功能。而服务提供者、集成者、服务中介和用户只获取服务的接口描述文档,Web服务对他们而言就是一种Web组件,因此他们可以使用组件测试的相关技术。

(3)非功能性测试

当请求一个服务时,服务使用者和服务提供者间应遵循服务级别协议(Service-level Agreement,SLA)。SLA是一种合法的合约,在合约中服务提供者列出了其对于某特定Web服务所保证的服务质量(QoS)的级别。它通常是服务提供者和用户经过协商以后达成的一致协议。它的一个重要功能是解决服务质量问题,服务若未能保证其期望的服务质量的情形都被视为对SLA协议的违背。因此,我们应加强SLA的测试。与此测试相关的QoS属性包括以下几个方面。

- 可用性(availability)。反映了Web服务做好准备(即可用)的可能性。较高的可用性意味着该服务更有可能在给定时间对一个请求进行处理。可用性常常用百分数来度量,例如一个Web服务的可用性可通过平均故障间隔时间和平均修复时间的百分率比值来表示。
- 可访问性(accessibility)。反映了能访问服务的用户群体和访问服务的便易程度。
- 完整性(integrity)。完整性描述了Web服务能保证其交易执行正确性的可能性。
- 性能(performance)。性能是反映Web服务质量的一个重要方面。主要包括两个因素:吞吐量(throughput)和响应时延(latency)。较大的吞吐量和较短的响应时延代表高的可靠性。吞吐量表示在给定时间内Web服务所处理的请求

的数量;响应时延表示服务对某个请求做出响应所花费的时间的长短。吞吐量越大,响应时延越短,性能就越好。

• 可靠性(reliability)。反映了 Web 服务即使是在系统或者网络出现故障的情况下能够正确运行并提供持续服务的能力。一个 Web 服务的可靠性通常是由每月或每年交易失败的次数来表示。

• 安全性(security)。包括诸如身份验证、加密和数字签名的技术和过程。一般来说,SLA 合约中定义了某特定 Web 服务所需的一定的安全性。因此,服务提供者就必须保证那个级别的安全性。

(4)回归测试

在 SOA 体系结构下,只有服务提供者掌握着服务的演化和升级情况,这给回归测试带来了很大的复杂性。我们知道,回归测试是在对软件做了修改后,为保证修改并未带来新的错误而进行的重新测试,这就需要服务的使用者和系统的集成者被告知服务是否发生变更,然而,在实际应用中,服务提供者并不一定知道有哪些用户正在使用该服务,所以提供者也就无法提醒用户服务的接口或是实现已经发生了改变。例如,在某个基于 Web 服务的应用的整个生命周期中,服务可能改变它自身的功能或 QoS 而保持服务的接口不变,这样的话就可能会影响整个应用的系统行为,这可以视为对 SLA 协议的违背。因此,所有使用该服务的应用都需进行回归测试。

从上面的讨论中我们可以看到对于不同的测试者,由于其角色(开发者、提供者、用户等)的不同,他们的测试目标和所采用的测试策略也有所不同。表 2 总结了不同的测试者执行不同测试策略的关键点。

表 2 从不同的角度讨论测试的相关策略

测试的策略 / 测试的视角	易测试性	功能测试	非功能性测试	回归测试
服务开发者	拥有服务的代码和服务的 WSDL 接口描述文档,丰富的测试信息	白盒测试/黑盒测试	测试环境不能反映提供者和用户的实际运行情况	了解服务变更情况,但不知有哪些用户使用该服务
服务提供者	WSDL 描述文件用于测试用例的设计	黑盒测试	测试是否保证 SLA 合约; 测试环境不能反映用户的实际运行情况	知道服务是否变更,但不知具体的变更细节
服务集成者	需要 WSDL 描述文件和服务提供者发布的测试用例组	黑盒测试	测试是否保证 SLA 合约; 难度大、代价高	不知道服务是否变更
UDDI 注册中心	需要 WSDL 描述文件和提供者发布的测试用例组	黑盒测试(检测注册服务是否满足应有的功能需求)	评估注册服务的质量;测试环境不能反映用户的实际运行情况	不知道服务是否变更
用户	需要 WSDL 描述文件和提供者发布的测试用例组以及用户自身设计的测试用例	黑盒测试(验证服务运行时的功能)	验证服务运行时的性能	验证服务演化后是否仍能满足自己的应用需求

3.3 Web 服务测试过程的组织框架

传统的应用测试中,为使软件测试走向规范,人们往往建立一个软件测试的管理体系,规范测试流程。该流程主要由下面 6 个阶段组成:测试规划、测试设计、测试执行、报告缺陷、修复缺陷、回归测试。

根据这一原则,Web 服务测试应遵循以下这些步骤,以保证高效的测试:

定义测试计划,概述测试过程;

从用例或业务需求中抽取测试用例;

为每个测试用例产生测试数据和/或测试脚本;

预先记录每个测试用例期望的结果;

执行测试用例;

验证测试用例的结果是否与期望的结果(测试预言)相匹配;

根据测试用例的执行结果生成测试报告以评估系统的质量,并将测试报告以简单易读的格式显示给测试者;

修复/解决系统中遗留的缺陷。

然而,Web 服务的分布式特征使得这一过程变得更加复杂。SOA 结构下的三种不同的角色包括服务提供者、服务中介和服务请求者,他们需要在一个分布式合作的环境下,共同完成以上的测试过程。如图 2 所示,测试通过测试控制代理(TestMaster)和多个测试代理(TestAgent)协作完成。对每个测试计划,TestMaster 都生成一组 TestAgents,由 TestAgent 通过 Internet 远程调用服务,同时它还负责与 TestAgent 的通讯,收集测试结果,以便做进一步的测试分析。

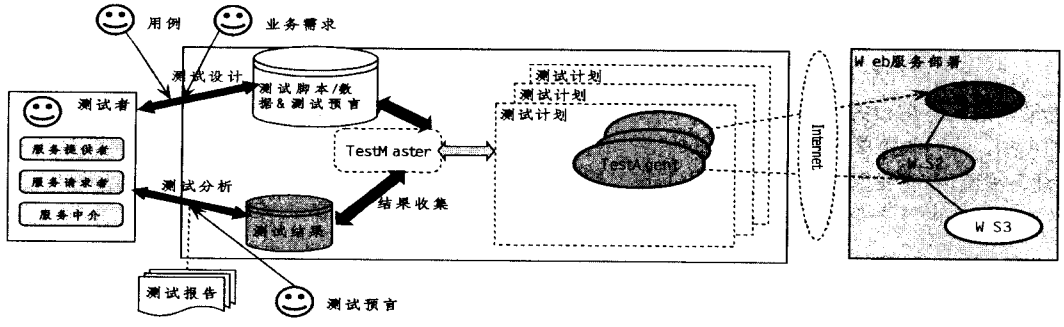

图 2 Web 服务测试过程的组织框架

4 Web 服务测试研究现状

随着网络应用从面向对象的架构向面向服务的架构方向转变,围绕这一新型应用模式的技术研究和应用越来越受到企业和个人的重视。其中,为提高服务可信性而进行的测试也引起了相关人士的关注。著名的调查分析公司 ZapThink

的分析家 B. De[15]指出了 SOA 体系下 Web 服务测试的 10 个主要问题，并将 Web 服务测试技术的研究分为以下三个阶段。

• 第一阶段：关注企业内部的测试，主要测试服务的基本功能，如功能、SOAP 消息和 WSDL 描述等。这是现阶段的 Web 服务测试技术；

• 第二阶段：关注面向服务的特征，主要测试 SOA 的发布、查找和绑定三个角色的能力，以及服务间的异步通讯问题和服务质量(QoS)问题；

• 第三阶段：关注服务的动态特性和集成的总体测试，如服务组合测试和版本测试。

目前，国内外对 Web 服务的测试研究已有一定的进展，但主要还停留在第一阶段。具体介绍如下。

(1)WSDL 扩展

文献[10]分别从输入/输出依赖、调用顺序、并发顺序规约和层次功能描述这四个方面对 Web 服务的 WSDL 文档进行了扩展，使得 WSDL 文档为测试提供了更多的信息，提高了 Web 服务的易测试性：输入/输出依赖标注了服务接口之间的联系，在回归测试中可以减少不相关测试，达到减少测试用例的目的；服务间的调用顺序给出了服务间的数据流和控制流的依赖关系；并发顺序规约用于描述多个服务的并发行为，这两者都便利了基于路径的测试用例的设计；层次功能描述是指采用层次化的结构形式对服务的功能进行描述，对基于功能分解的测试用例的设计很有帮助。

(2)基于合约式设计(DbC)的 Web 服务测试

合约是服务提供者和服务请求者管理两者之间交互的一组规则，它表明了两者间应遵守的义务和责任。常见的合约类型包括：前置条件、后置条件和不变式。请求者只有在满足前置条件的条件下才能调用对应的方法；提供者承诺当操作结束时，后置条件指明的任务将被完成，并且不变式仍然满足。

文献[6]的思想是服务提供者和服务请求者两端都引入合约式设计[11]的概念，并且使用可视化图形的方式描述合约的规则：服务提供者提供供应合约(provided contract)，其图形表示的左边代表服务的前置条件，右边表示服务的后置条件；服务请求者提供请求合约(required contract)，其图形表示的左边表示请求者将向服务提供的信息，右边表示他在使用服务后达到的状态。这样，一方面可通过供应合约生成测试组(测试组通常与测试目标相对应，它由多个测试用例组成)：测试用例的输入由供应合约的前置条件中产生，测试预言则根据供应合约的后置条件确定；另一方面，请求合约可用于组件的模拟：有时测试可能局限于单个的 Web 服务，为模拟被测试服务请求的服务的行为，则使用相应的请求合约进行服务行为的模拟。每当被测服务使用请求合约请求其它服务时，测试驱动将被调用，它根据客户端服务提供的输入数据应用请求合约产生对应的一个返回值，从而达到模拟请求服务的目的。

(3)变异测试

变异测试起初是针对白盒测试提出的一种排错性测试技术，它的基本原理是使用变异算子对软件进行作用，从而产生许多不同的软件版本，称为“变异体”。当变异体的运行结果不同于原程序的运行结果时，称该变异体被杀死。当执行完所有的变异体后，测试者将获得两方面的信息：1)被杀死的变异体数目；2)未被杀死的变异体，其在功能上等价于源程序，称为等价变异体。变异测试使用变异充分度(被杀死的变异体数目与非等价的变异体数目的比值)来衡量测试输入集的充分性程度。

文献[12]将变异测试的思想应用于 Web 服务的接口描述文档——WSDL 文档。在 WSDL 文档中主要使用〈types〉、〈message〉、〈portType〉和〈binding〉这四种基本元素来描述 Web 服务的接口信息。文中阐述了三种类型的变异算子：Switch(用于交换元素的顺序)、Special(改变元素的取值)和 Occurrence(添加或删除元素的出现频率)，而变异算子的操作对象则是以上提到的四种基本元素，这样则生成了不同类型的变异算子：STCE，STCA，OTCE，OTCA，STEN，STEN，STSE，STSA，SMP，SPM。这些变异算子的使用在一定程度上能暴露 WSDL 接口的错误和 Web 服务本身的一些逻辑错误。

(4)基于 GT 规则的服务注册前的测试

目前，服务的描述和发现在很大程度上是基于服务操作的签名和简单的服务分类，因此很难保证 UDDI 中心返回的服务能以客户期望的方式运作。为了实现高质量的服务发现机制，文献[13]提出了一种基于 GT 规则(Graph Transformation Rule)的服务注册前的测试。首先服务提供者提供 WSDL 接口文档和服务的行为规约文件(一个 Web 服务通常具有若干的产品规则，这些规则反映了服务在不同的状态使用不同的输入数据执行以后可能发生的计算操作。服务的行为规则由 GT 规则来表示)。接着，服务提供者向 UDDI 注册中心发布该服务的 WSDL 文件和行为规约文件，注册中心根据 GT 规则自动生成测试组并通过远程调用服务执行具体的测试用例，最后根据返回结果判断测试是否通过。只有通过测试的服务才允许正式在 UDDI 处发布。

(5)基于 PSM 的 UDDI 处的互操作性测试

互操作性表明了软件与其它系统交换数据和服务的难以程度。SOA 体系结构下，通常涉及服务提供者、服务请求者和服务中介等角色间的交互，测试它们之间的互操作性是 Web 服务测试的一个重要方面。然而，由于 Web 服务在发布时只提供其接口信息而不包括其实现细节，这在一定程度上降低了提供者和请求者的耦合度，但却为测试带来了一定的困难。

文献[14]指出服务提供者不仅提供服务的接口信息，还提供与该服务交互时必须遵守的协议。文中使用 UML2.0 中新近引入的协议状态图(PSM)来描述服务的协议。在服务提供者将服务的 WSDL 文档和相应的 PSM 规约提交给 UDDI 注册中心后，注册中心将该服务放入“尚待决定”的数据库中，然后创建 WS tester 并根据提供的两份规约文件对该服务进行测试，在测试过程中，监控待注册的服务与已注册的服务之间的交互情况，通过检测待注册服务对其它服务的调用顺序是否正确来验证该服务是否能正确地同其它服务进行交互，从而决定该服务是否被批准注册。

(6)加强的 UDDI 验证中心

Tsai 等人在文献[8]中提出了使用一种加强的 UDDI 服务器来验证 Web 服务的策略。UDDI 注册中心的测试主要包括检入和检出两种：1)检入测试(Check-in Test)：参与的角色包括 UDDI 中心和服务提供者。在服务被提交到 UDDI 中心之前，注册中心的测试引擎 TestMaster 根据本地的测试数据库，对提交的服务首先进行服务质量的检测。通过测试的服务成功地被注册到 UDDI 中心处，而不合格的服务则被拒

绝注册。2)检出测试(Check-out Test):参与的角色包括 UDDI 中心和服务请求者。UDDI 中心在接收服务请求后,首先 TestMaster 通过检查接口和版本信息查询 UDDI 目录中符合要求的服务,并对候选的服务进行回归测试。检出测试在以下两个方面发挥作用:①对满足用户需求的多个候选服务进行评估(这里采用了 progressive group testing 技术),选取最符合用户需求的服务;②已注册的服务有可能发生新的变更,UDDI 中心须确认服务是否改变和变更后对服务质量所造成的影响。

(7)基于场景的 Web 服务测试

场景技术是在软件开发中用来捕获软件需求和系统功能的方法,它是软件体系结构建模的主要依据,并可以用来指导测试用例生成。场景从用户的角度描述了系统的行为,反映了系统期望的运行方式。场景关注的系统外部的行为而并不考虑系统的内部实现细节,它已经成为一种重要的测试设计技术。

Tsai 等人在文献[22]中从服务请求者和 UDDI 服务中介两个角度考虑 Web 服务的测试,提出了一种使用多层次场景描述的方法来描述系统的行为:1)首先抽取单个子系统的场景;2)其次,描述多个系统间的交互情况;3)最后根据系统间的交互情况产生整个系统的场景。这些不同抽象级别的场景将用于测试脚本的生成。基于场景的测试过程分为下面 3 个过程:1)setup 阶段:将被测服务转到场景的前置条件中指定的状态,并产生场景中指定的输入事件;2)execution 阶段:使用适当的方法参数远程或本地地调用场景中指定的方法。3)verification 阶段:使用场景相关的验证模式对测试进行最后的验证分析工作。

该测试方法支持 UDDI 中心处的检入和检出测试、单个服务的测试、面向服务体系结构的测试和动态运行时测试。

(8)基于提供者发布测试组的服务测试

Bruno 及其研究人员在文献[7]中提出了一种将测试用例作为合约的一部分进行发布的策略方便了用户进行回归测试。首先由服务提供者给出一个测试规格(规范是以 XML 形式提供),其中描述了服务相关的测试用例集和 QoS 断言集。测试规格连同服务接口文档一同被发布,当用户请求服务时将下载相应的测试组,用户定期运行该测试组,验证服务是否仍然满足原有的功能和非功能性需求。如果在这段期间服务发生演化(如版本升级),那么服务的行为将可能造成提供者/请求者合约的违背。

(9)基于协议的 Web 服务交互测试

Web 服务的通信协议采用标准的 SOAP 协议,SOAP 通信消息是基于 XML 技术的,具有良好的可读性和易分析性,因此通过对协议的交互内容进行分析是 Web 服务交互测试的可行的测试方法。文献[24]提出将被测服务视为可控制中心,被调用的其它服务视为黑盒,截获 Web 服务间的 SOAP 通信消息,将消息表示为知识库,并利用规则库对消息进行重演、控制和分析。为此,文中提出了 Web 服务交互测试的概念模型,它分为三个层次:数据获取层截获服务间的通信信息,并将其传送给数据管理层;数据管理层对通信消息进行预处理后,由数据分析层描述服务间的调用关系,实现通讯状态的分析控制、通讯过程的重演以及进行 Web 服务的压力/负载测试。

(10)基于消息流的 Web 服务集成测试

文献[21]中,S Dustdar 和 S Haslinger 通过基于分析消息流来测试服务及服务端点间的消息流的理论,提出了一种服务集成测试工具 SITT 的思想及其原型实现。工具的使用代替了繁琐耗时的人工测试方法,提高了测试的效率和正确性。此外,测试可在夜间进行,因此它并不影响白天的开发工作。

SITT 中使用一个控制代理(MA)、多个测试代理(TAs)以及测试后台程序(Test Daemon)协作完成服务的工作流测试:

• 测试代理(Test Agent,TA):是与服务端点运行在同一主机上的程序,它们的主要作用是解析服务端点生成的日志文件,并将该日志中的信息传送给控制代理(MA)。

• 控制代理(Master Agent,MA):MA 主要负责收集各个服务端点生成的日志文件,并将接受到的信息存储于测试数据库中。

• 测试后台程序(Test Daemon):由 MA 将接收到的信息存于数据库后,每隔一段特定的时间(通常是几秒),测试后台程序从测试数据库中提取信息,根据预先定义的测试描述文件(该文件用 xml 语言编写)分析该次测试是否成功。与此同时,它还可以对测试结果做一些统计工作:如系统的可靠性、性能、吞吐量等。

该测试工具可用于测试工作流的执行是否符合系统设计者或测试人员的定义,也可用于系统的统计分析,如消息时序、系统性能、吞吐量和系统的可靠性等等。自动测试工具的使用代替以往反复地、手工的测试方式,将有效降低测试复杂 SOA 系统的复杂性,为测试小组更好、更快地测试提供了条件。它的优势在于服务端点可为任何的服务适配器而不仅仅局限于 Web 服务。并且,该工具的后续开发中将实现通过简单的重配置过程将其转变为系统的监控工具,有效地将测试和监控的方法结合起来,提高了服务的可信性和服务质量。

(11)群测试

Web 服务的动态绑定的特性给服务带来了许多不确定性:新的服务在运行时可由现存的 Web 组合而成;并且在一个组合服务中,任何组成服务在执行时一旦失效则需要被其它的服务的单个服务或组合服务动态地替代。这些使得组合服务的测试变得更加困难。文献[17]提出使用群测试的技术来测试服务的组合:假设一个组合服务包含 n 个单元 Web 服务集并且 Set_i 具有 m_i 个等价的单元服务:$Set_i=\{S_{i1},S_{i2},\cdots,S_im_n\}(1\leqslant i\leqslant n)$,这样将有 $m_1 * m_2 * \cdots * m_n$ 种可能的组合情况。群测试的原理是:设组合服务 $WS_1=(S_{11},S_{21},\cdots,S_{n1}),\cdots,WS_k=(S_{n1},S_{n1},\cdots,S_nm_n)$,组合服务的输入被广播到所有的组合服务 $WS_1,WS_2,\cdots,WS_k$,它们的输出被直接传递给一个"票决服务",该服务基于一种权重票决算法选择得到多数投票的服务组合(这里的权重是由服务的可靠性历史所决定)。每个服务的输出与多数投票的服务的输出进行比较。结果不一致的发生次数将被记录到错误日志中并被用于评估 Web 服务的可靠性级别、测试脚本的有效性和生成每个测试用例的预言。

各种方法的比较见表 3。

5 总结

5.1 存在的问题

对于 Web 服务的测试,一个很大的困难在于服务是被调用的而非源代码可知的。在这种情况下,除服务开发者外,人们采用的服务的测试方式只能是黑盒的,即通过服务的 WS-

表3 Web服务测试方法的比较

编号	测试技术	适用的测试角色	测试内容	是单个服务测试还是多个服务的测试	有无工具支持
1	WSDL扩展	所有	易测试性设计	单个服务	无
2	基于合约式设计(DbC)的Web服务测试	服务提供者、服务集成者和用户	功能测试	单个服务	Jtest tool by Parasoft, JContract
3	变异测试	服务提供者、服务集成者和用户	功能测试	单个服务	无
4	基于GT规则的服务注册前的测试	UDDI注册中心	功能测试	单个服务	无
5	基于PSM的UDDI处的互操作性测试	UDDI注册中心	互操作性测试	多个服务	无
6	加强的UDDI验证中心	UDDI注册中心	检入测试、检出测试	单个服务、多个服务	TestMaster, TestAgents和TestMonitors
7	基于场景的Web服务测试	用户、UDDI注册中心	功能测试、非功能性测试(同步异步能力、可用性、可访问性、完整性、性能/吞吐量、可靠性、规范性和适应性)	单个服务、多个服务	TestMaster, TestAgents和TestMonitors
8	基于提供者发布测试组的服务测试	用户、服务提供者、服务集成者和UDDI注册中心	功能测试、非功能性测试、回归测试	单个服务	Test case generation tool & Test suite runner
9	基于协议的Web服务交互测试	服务集成者、用户	功能测试、非功能性测试(压力/负载测试)	多个服务	Web软件交互测试系统,开发中(已实现部分功能)
10	基于消息流的Web服务集成测试	服务集成者、用户	功能测试、非功能性测试(性能/吞吐量、可靠性等)	多个服务	SITT
11	群测试	服务集成者、用户和UDDI注册中心	非功能性测试(可靠性)	单个服务、组合服务	WebStrar[23]

DL描述文档来获得相关的测试信息。为了对服务进行更加充分的测试,除了对WSDL进行相应的扩展外,还应考虑如何最大程度地获得更丰富的测试信息。一种普遍的思想是由服务提供者提供必要的Web服务交互规则、产品规则或测试组,方便服务用户进行测试,但是到目前为止还没有形成统一的标准。此外,为了保证高可信性的Web服务,增强UDDI注册中心的验证能力的方案也吸引了越来越多的学者的关注,然而这些方法通常都需要扩充的Web服务规约,且需要第三方认证中心的认可和支持。目前各种方法的本质还是借鉴传统的测试技术,如变异测试、基于合约式设计的测试、基于服务提供者发布测试组的测试(与构件测试技术类似)等。这些方法中有许多都是在原有的基础上根据Web服务的特点进行改进而得到的。这些研究中大多数是对单个服务的测试问题进行了探讨,而服务的动态组合的测试和服务集成的总体测试等方面的研究却很薄弱,有待进一步加强。

5.2 未来研究方向

Web服务和SOA体系结构的出现是Internet应用发展的一个重要的里程碑。据Gartner Group预测,到2008年,SOA将成为占有绝对优势的软件工程实践方法,它将很可能结束传统的整体软件体系架构长达40年的统治地位,届时将有70%的企业在进行IT建设时会转向SOA。随着SOA结构在企业应用中的比重越来越大,考虑到服务的可信性是其成功的关键,作为保证服务质量和可靠性的重要手段,测试将受到人们越来越多的重视。然而,由于SOA特殊的结构特点,与传统软件测试相比,Web服务测试将面临许多新的挑战。目前,关于Web服务测试的方法、技术和工具研究还存在很大的不足,作者认为以下几个方面是未来研究的关键方向。

·建立完整的Web服务测试过程模型。如在分布式环境下,通过使用多种不同分工的测试代理协作完成测试任务。这样,整个测试过程就能得到高效的、系统的组织和管理。

·测试与监控相结合。由于测试是在系统真正被部署使用之前采用的系统验证的方式,它不能反映系统运行时的动态性和适应性调整,而系统监控正好弥补这一不足。因此,将测试与监控两者结合起来检测系统不失为一种很好的验证策略。

·服务间通信信息的捕获-分析和重演。这些属于测试模拟的范畴。它的基本思想是首先截获系统调用过程中所用的通信信息;根据具体的需要对数据进行净化与抽取;最后,依据特定的数据模型,对系统的通讯状态进行分析控制,还可以重演通讯过程和进行压力负载测试。

·测试自动化技术和工具研究。使用自动化测试工具代替繁琐的、耗时的、易出错的人工测试,能够迅速、彻底地对系统进行测试,从而提高软件质量,节省开支,缩减产品开发周期,极大地提高测试的效率。

参考文献

[1] Castro-Leon E. A perspective on Web Services [EB/OL]. http://www.webservices.org/index.php/article/113/1/61/,2004

[2] Tsalgatidou A, Pilioura T. An Overview of Standards and Related Technology in Web Services[J]. Distributed and Parallel Databases, 2002, 12 (2/3): 135

[3] Booth D, Haas H, Mccabe F, et al. Web Service Architecture [EB/OL]. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/wsa.pdf, 2004

[4] W3C Web Services Activity. http://www.w3.org/2002/ws/

[5] Voas J M. Software Testability Measurement for Assertion Placement and Fault Localization// Proceedings of the Automated and Algorithmic Debugging. Saint-Malo, 1995: 133-144

[6] Heckel R, Lohmann M. Towards Contract-based Testing of Web Services. Theoretical Computer Science, January 2005, 116: 145-156

[7] Bruno M, Canfora G, Penta M D, et al. Using Test Cases as Contract to Ensure Service Compliance across Releases // Proceedings of the 3rd International Conference on Service Oriented

Computing (ICSOC 2005), LNCS 3826. Springer, 2005: 87-100
[8] Tsai W T, Paul R, Cao Z, et al. Verification of Web services using an enhanced UDDI server // Proceedings of IEEE WORDS. 2003: 131-138
[9] Tsai W T, Paul R, Song W, et al. Coyote: An XML-based Framework for Web Services Testing // Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). Tokyo, Japan, 2002: 173-174
[10] Tsai W T, Paul R, Wang Y, et al. Extending WSDL to Facilitate Web Services Testing // Proceedings of 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). Tokyo, Japan, 2002: 171-172
[11] Meyer B. Object-Oriented Software Construction. 2nd Ed. Prentice-Hall, Englewood Cliffs, NJ 07632, USA, 1997. http://www.prenhall.com/allbooks/ptr_0136291554.html
[12] Siblini R, Mansour N. Testing Web services // Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications. 2005: 135
[13] Heckel R, Mariani L. Automatic conformance testing of web services // Proceedings of FASE. Edinburgh, Scotland, Apr. 2005: 2-10
[14] Bertolino A, Polini A. The audition framework for testing Web services interoperability // Proceedings of IEEE Computer Society on EUROMICRO-SEAA. 2005: 134-142
[15] De B. Web Services-Challenges and Solutions. WIPRO white paper, 2003. http://www.wipro.com
[16] Foster H, Uchitel S, Magee J, et al. Model based verification of web service compositions // Proceedings of the 18th IEEE International Conference on Automated Software Engineering. Montreal, Quebec, Canada, 2003: 152-161
[17] Tsai W T, Chen Y, Paul R, et al. Cooperative and grou Ptesting in verification of dynamic composite Web services // Proceedings of the 28th Annual International Computer Software and Applications Conference. Hong Kong, China, 2004: 170-173
[18] Li Y, Li M, Yu J. Web Services Testing, the Methodology, and the Implementation of the Automation-Testing Tool // Grid and Cooperative Computing: Second International Workshop, GCC 2003. Shanghai, China, December 2003
[19] Tsai W T, Chen Y, Cao Z, et al. Testing Web Services Using Progressive Group Testing // Lecture Notes in Computer Science, 2004, 3309: 314-233
[20] Xu W, Offutt J. Generating Test Cases for Web Services Using Data Perturbation. ACM SIGSOFT Software Engineering Notes, 2004, 29(5): 1-10
[21] Dustdar S, Haslinger S. Testing of Service Oriented Architectures- A practical approach // Proceedings of the 5th Annual International Conference on Object-Oriented and Internet-Based Technologies, Concepts, and Applications for a Networked World. NODe 2004, Erfurt, Germany, September 2004
[22] Tsai W T, Paul R, Yu L, et al. Scenario-based web service testing with distributed agents. IEICE Transaction on Information and System, 2003, E86-D (10): 2130-2144
[23] Tsai W T, Zhang D, Chen Y, et al. A Software Reliability Model for Web Services // Proceedings of the 8th IASTED International Conference on Software Engineering and Applications. Cambridge, MA, November 2004: 144-149
[24] 黄宁，余莹，张大勇. Web服务软件测试技术的研究与实现[J]. 计算机工程与应用，2004(40)：147

(上接第 244 页)

中心化过程中，采用的是整体能量优化方法，因此仍可能导致最终 Snake 上某些局部点并不处于能量最低位置，这就是 B-Snake算法所求的骨架中心性仍然有一定误差的原因。改进方面可考虑在用控制点进行 Snake 的演化之后，再用 Snake 上结点(Knots)进行一次演化。

参 考 文 献

[1] Zhou Y, Thompson P, Toga A W. Extracting and representing the cortical sucli [J]. IEEE Computer Graphics and Applications, 1999, 19(3): 49-55
[2] He L, Han C Y, Everding B, et al. Graph matching for object recognition and recovery [J]. Pattern Recognition, 2004, 37: 1557-1560
[3] Shamir A, Shaham A. Skeleton based solid representation with topology preservation[J]. Graphical Models, 2006, 68: 307-321
[4] Fuhrer M, Jensen H W, Prusinkiewicz P. Modeling hairy plants [J]. Graphical Models, 2006, 68: 333-342
[5] 田绪红，李志垣，韩国强，等. 基于切面算法的三维植物根系图象骨架生成方法[C] //第十二届全国图象图形学学术会议论文集. 北京，2005：655-658
[6] 朱同林，方素琴，李志垣，等. 基于图像重建的根系三维构型定量分析及其在大豆磷吸收研究中的应用[J]. 科学通报，2006，51(16)：1885-1893
[7] Deschamps T, Cohen L D. Fast extraction of minimal paths in 3D images and applications to visual endoscopy[J]. Med Phys, 2001, 5(4): 281-289
[8] Ma C M, Wan S Y, Chang H K. Extracting medial curves on 3D images[J]. Pattern Recognition Letters, 2002, 23: 895-904
[9] Paragyi K. A 3-subiteration 3D thinning algorithm for extracting medial surfaces[J]. Pattern Recognition Letters, 2002, 23(6): 663-675
[10] Amiya A, Saito M. Thinning by curvature flow[J]. Journal on Visual Communication and Image Representation, 2006, 17: 27-41
[11] Wong W T, Shih F Y, Su T F. Thinning algorithms based on quadtree and octree representations[J]. Information Science, 2006, 176: 1379-1394
[12] Niblack C W, Gibbons P B, Capson D W. Generating skeletons and centerlines from the distance transfrom[J]. Computer Vision, Graphics, and Image Processing, 1992, 54(5): 420-437
[13] Gagvani N. Skeletons and volume thinning in visualization[D]. MS thesis. Dept. of Electronical and Computer Eng., Rutgers Univ., New Brunswick, N. J., June 1997
[14] 樊雅萍，黄生学，温佩芝，等. 基于数字距离变换的3D模型骨架提取算法[J]. 信息与控制，2004，33(6)：685-693
[15] 张若文，滕奇志，孙晓刚，等. 一种快速简便的图像骨架变换方法. 信息与电子工程，2003，1(1)：1-5
[16] Zhou Y, Toga A W. Efficient skeletonization of volumetric objects[J]. IEEE Transaction on Visual Computer Graph, 1999, 5(3): 196-209
[17] Paik D S, Beaulieu C F, Jeffrey R B, et al. Automated flight path planning for virtual endoscopy[J]. Med Phys, 1998, 25(5): 629-637
[18] Maddah M, Afzali-Kusha A, Soltanian-Zadeh H. Efficeient centerline extraction for quantification of vessels in confocal microscopy images[J]. Med Phys, 2003, 30(2): 204-211
[19] Maddah M, Soltanian-Zadeh H, Afzali-Kusha A. Snake modeling and distance transform approach to vascular centerline extraction and quantification[J]. Computerized Medical Imaging and Graphics, 2003, 27: 503-512
[20] Kass M, Witkin A, Terzopulos D. Snake: active contour models [J]. International Journal of Computer Vision, 1988, 1: 321-331
[21] Klein A K, Lee F, Amini A A. Quantitative coronary angiography with deformable spline models[J]. IEEE transactions on medical imaging, 1997, 16(5): 468-481
[22] Tian Xuhong, Han Guoqiang, Chen Maozi, et al. Skeleton- based surface reconstruction for visualizing plant roots[C] // Proceedings of 16th Internatinal Conference on Artificial Reality and Telexistence-Workshops. Hangzhou, 2006: 328-332