基于Web服务生命周期的测试技术研究进展

李雯睿 王志坚 毛莺池
(河海大学计算机及信息工程学院 南京 210098)

摘 要 Web服务是一种新的分布式计算范型——面向服务的体系结构的具体实现之一。然而,Web服务的松耦合、动态性和可组合性等特点给Web服务测试带来了极大的挑战。如何寻求高效的Web服务测试技术和开发实用的测试工具,是当今软件业界一个亟待解决的课题。当前Web服务的测试方法已经贯穿于服务的开发、预发布、运行以及演化的整个生命周期。提出了从Web服务生命周期的角度分析Web服务测试的重要参数,在此基础之上归纳和比较了近年来出现的一些典型的测试方法和技术,最后展望并探讨了Web服务测试今后的研究方向。

关键词 Web服务,Web服务生命周期,测试
中图法分类号 TP311.5 **文献标识码** A

Research Progress in Testing Techniques Based on Web Service Life Cycle

LI Wen-rui WANG Zhi-jian MAO Ying-chi
(College of Computer and Information Engineering, Hohai University, Nanjing 210098, China)

Abstract Web service is a new distributed computing paradigm, which is one of the implementations of Service-oriented Architecture. However, the new features of Web service, such as loose-couple, dynamics and compositionality, bring great challenges to test Web service. How to find efficient and effective testing techniques of Web service and to develop practical test tools is a promising research theme to be solved. Currently, testing methods have been applied through the whole life cycle of Web service which consists of development, pre-publication, runtime and evolution. Therefore, from the view of Web service life cycle, the paper first analyzed several important parameters effecting testing techniques of Web service. Then the state-of-the-art testing methods and techniques were surveyed and compared in terms of these parameters. Finally, the directions of future research were explored.

Keywords Web service, Web service life cycle, Testing

1 引言

面向服务的计算(Service Oriented Computing, SOC)是一种全新的计算模式,以Web服务为基础,有效解决在分布、动态、异构环境下的分布式应用集成问题。Web服务作为一种自治的、与平台无关的网络化构件,具有松耦合、支持动态组合和重配置等特点,从而为Internet环境中的软件开发和分布式应用提供了新的技术。Web服务采用面向服务的体系结构(Service-Oriented Architecture, SOA),通过XML, SOAP, WSDL和UDDI等标准协议描述、发布和发现服务,实现跨平台、跨组织的服务之间的互操作和集成。

Web服务具有如下特点:①利用标准协议开发服务,系统有良好的互操作性;②动态地发现和集成服务;③由于服务可被替换,动态演化速度快,具有更好的接口兼容性和过程一致性;④可实现系统的重用性和可扩展性。

为了提供可信任的Web服务,测试是保证Web服务质量的重要途径之一,因此Web服务测试已经成为学术界和业界关注的热点研究领域。但是Web服务固有的新特性给测试带来了一系列的问题:由于源代码对测试者通常是不可见的,测试者只能采用黑盒测试技术;如何评估测试充分性、准则的有效性;离线测试时如何仿真服务的实际运行环境;当服务部署后,如何从实际运行环境中有效地选择和分析收集的运行数据,而不会增加系统开销;集成的服务构件是由不同的参与者开发和管理的,运行时环境不可预期地发生改变,而集成者不能完全控制这些服务构件,所以需考虑这种情况下如何实施测试;当服务动态地演化,如何减少重测试的工作;需考虑对可靠性、健壮性和性能等QoS属性实施测试,以提供可信任的服务。由此可见,传统的软件测试技术已经不再适用于Web服务。因此,研究界与业界提出了一些新方法和技术来测试Web服务,从而为服务质量提供了有力的保障。

调查发现,现有的Web服务测试方法已经贯穿于服务的开发、预发布、运行以及演化的整个生存周期。本文试图从

到稿日期:2008-07-09 返修日期:2008-10-08 本文受国家自然科学基金(60573098),国家863计划重点项目(2007AA01Z178);国家973子课题(2002CB312002)资助。

李雯睿(1981—),女,博士研究生,CCF学生会员,研究方向是分布式计算、Web服务组合,E-mail: lwr2255@yahoo.com.cn;王志坚(1958—),男,博士,教授,博士生导师,CCF高级会员,研究方向是软件复用、构件技术、分布式计算;毛莺池(1976—),女,博士,讲师,研究方向为分布式计算、Web服务。

Web服务生命周期的角度，综述在生命周期的不同阶段测试Web服务采用的方法和技术。第2节给出了Web服务测试涉及的重要参数。第3—5节在此基础之上分析和归纳了近年来出现的一些典型的Web服务测试方法和技术，并指出这些方法的优缺点。最后讨论了Web服务测试今后的研究方向。

2 基于Web服务生命周期的测试

2.1 基于Web服务生命周期的测试阶段

Web服务具有开放的标准接口、可发布、动态发现和快速组合等特征，这些特征改变了整个软件生命周期(包括需求分析、设计、实现、验证和确认、运行和维护)，因此将Web服务的生命周期划分为3个阶段：开发阶段、预发布阶段和运行阶段。

在Web服务生命周期的哪个阶段实施测试？一般来讲，移除错误的成本是随着生命周期的进展而增加的，所以在生命周期早期要尽可能地发现潜在的错误，以减少生命周期后期测试所带来的昂贵的成本开销。测试时机依赖于执行环境，开发者在开发阶段离线测试Web服务，然而并不能预见服务实际的运行环境，所以需要其他参与者在服务发布前和运行时的实际执行环境中充分测试服务。对应3个测试阶段：离线测试阶段、在线审计阶段和在线监控阶段。

离线测试阶段：在离线环境中测试已开发的服务，以提供满足规约的服务。为了得到可靠的结果，需要尽可能地仿真实际的运行环境。

在线审计阶段：在服务注册前测试服务，确认并评估服务的互操作性，其执行环境是在线注册时Web服务的运行环境。

在线监控阶段：在实际的运行环境中观察服务行为，以发现与期望行为可能不一致的行为。在线审计阶段与在线监控阶段的区别是，部署的服务是否已发布并提供给终端用户。

需要指出的是，Web服务测试涉及的参与者包括服务开发者、部署者、UDDI注册中心、集成者、第三方和用户。这些参与者测试目的不同，采用的测试技术也不同。除了服务开发者可使用白盒测试技术，其他参与者均使用黑盒测试技术，这是由于服务的设计和实现细节对这些参与者是不可见的。

2.2 Web服务测试参数

根据每个方法针对解决的问题，需要从以下几个方面考虑对一些典型的Web服务测试方法进行分析和归类。

(1)组合服务的协作模型

组合服务的协作模型是指采用哪种建模语言组合一些服务，以提供新的、功能更强大的组合服务，主要有以下3类。

编排服务(Orchestration Service)：从局部的观点由中心控制者来完成服务组合。根据服务部署者提供的WSBPEL[1]规约，验证编排服务之间是否正确地被调用。

编舞服务(Choreography Service)：从全局的观点以对等的方式组合服务。编舞服务关注的是多个参与者之间的协作，没有中心控制者，验证的是编舞服务的角色与WS-CDL[2]规约是否一致。

基于语义的组合方式：以OWL-S[3]为例，通过共享本体实现Web服务间的自动发现、组合和监控。

(2)属性类型

属性类型包括功能属性和非功能属性。

功能属性：根据服务的规约选取测试用例，以检验输入输出信息是否完全符合规约中有关功能需求的规定。

非功能属性：与服务质量相关的属性，是以量化的方式度量的，包括性能、可靠性、健壮性、可用性和安全性等。

3 离线测试

离线测试是在Web服务生命周期的开发阶段对Web服务进行测试。其目的是在仿真的环境中，测试已开发的服务是否满足开发者定义的规约。就离线测试阶段而言，采用的方法和技术各有利弊，它们的差异体现在测试技术的选择。对这些方法的选取需要根据具体问题来决定，也取决于测试者的个人经验。

3.1 基于传统方法扩展的技术

3.1.1 变异分析

变异分析是一种排错性测试技术，用来评估测试用例集的充分性。其基本思想是，手工或自动地将错误引入源程序中，产生多个错误的版本，也就是将变异算子引入到源程序中，该错误版本称为变异体。当变异体的输出与源程序的输出不匹配时，变异体就被杀死。测试用例无法将变异体杀死，就称该变异体在功能上等价于源程序，这类变异体称为等价变异体。这里用变异充分度来衡量测试用例集的充分性。

为了提高测试效率和降低测试成本，Jiang等人[4]在WSDL文档中定制服务提供者和请求者的合约，并提出一种合约变异技术自动生成测试用例。采用等价类划分和边界值分析方法生成初始测试用例集，然后定义合约变异算子，作用在WSDL文档产生合约变异体，比较输出结果和合约变异预言(oracle)。如果比较结果不同，则测试用例杀死该合约变异体，最后从初始测试用例中选择达到一定合约变异充分度的测试用例。该方法实现了一个Web服务的测试用例自动生成原型工具WSTDGT，其缺点是不能处理复杂的数据类型，缺乏理论证明合约变异算子是否完备。

3.1.2 基于模型的测试方法

软件系统的发展趋势越来越复杂，而测试成本也逐渐升高，传统的测试技术已不再适用于测试Web服务。在开发阶段使用基于模型的测试方法会有效避免这些问题。该方法是一种黑盒测试技术，作为一种测试Web服务的重要途径，近年来备受研究者的关注。

由于WSDL隐藏了服务内部的实现细节，只描述了调用的静态方面，缺乏语义和行为信息。针对这样的问题，Frantzen等人[5]利用符号转换系统(Symbolic Transition Systems，STS)对Web服务复杂的交互行为建模，规定了规约模型和实现模型之间的实现关系(Implementation Relation)。根据实现关系和模型，自动生成测试用例集，并执行测试过程，最后判断Web服务实现与其规约是否一致。该方法支持同步和异步通信，但未考虑操作之间的相关性。类似地，文献[6]提出EFSM(Extended Finite State Machine)来表示行为信息，自动地从EFSM模型中生成测试用例，检验服务行为的正确性。EFSM适合用于描述服务行为，能够更为精确地表示服务行为，准确检测到潜在的错误。该方法的缺点是生成测试用例的开销过大。

3.1.3 分布式实验

离线测试时，为了获得有效的结果，实验环境必须仿真服务实际的执行环境。许多参数影响服务的行为，这些参数在实验中需要反复配置。但由于使用场景、环境条件或操作上下文的不同，配置服务所有可能的运行环境是很困难的，且代价高、耗时。另外，考虑到服务的规模、异构性和动态性等特点，使得难以手工引导实验。还有，使用服务的场景不可预期地快速改变，所以实验框架必须有效地支持重新配置实验。因此，需要提供给软件工程师工具和方法，反复地评估设计和过程，允许他们通过自动的、反复的实验快速地评估服务设计和实现。

文献[7]的目的是在开发阶段的后期模拟实际的执行环境，提供可执行的原型，反复运行实验，产生精确的结果。开发的框架 Weevil 能够使实验过程自动化，包括负载生成、实验部署和运行、数据收集。当实验配置参数改变时，Weevil 能够快速地重新运行实验，验证模型的通用性、负载逼真度、框架的可扩展性，并量化自动化程度，从而减轻测试者需要创建和维护大量的实验控制脚本的工作，并减少了测试开销。然而，该框架还需考虑支持动态负载，需提供给用户一种方式来规定负载修改规则、响应/请求关系，并将负载信息反馈给实验过程。

3.2 性能和健壮性测试

服务设计过程是否有效，不仅要考虑测试功能属性，还要考虑测试非功能属性。这里主要考虑保证其性能和健壮性。

3.2.1 性能测试

组合服务对系统性能有不可预期的影响。Grundy 等人[8]扩展了性能测试床生成器 MaramaMTE，评估组合服务的性能。首先利用可视化的服务组合规约语言 BPMN 或 ViTABaL-WS 快速地实现服务组合，然后用 MaramaMTE 建模更为详细的体系结构视图，接着从 MaramaMTE 中生成服务桩代码，并开发客户端负载模型，最后从组合模型和负载模型中生成一个或多个测试床。执行测试床、压力测试服务组合，并收集结果，发送给工程师，以分析可视化的性能结果，修改组合、负载模型和测试描述，重新执行性能测试来比较不同的性能描述。当组合快速改变时，建模方法的简化和测试床的自动生成使得测试能够确定优化配置，重新实施测试不会带来太多的编程开销。该方法的不足之处：只是初步实现测试性能功能(包括吞吐量和服务组合的响应时间)，需要更充分地反映历史使用数据和服务器的响应请求；只支持测试静态的服务组合的性能，还需考虑组合服务动态改变时如何实施性能测试。

考虑到评估协作场景中组合服务(编排或编舞)的 QoS 属性，Bertolino 等人[9]实现了 Puppet 工具。根据提供服务的 WSDL，WS-CDL 或 WSBPEL，WS-A 和 SLA 规约，自动生成性能测试床，并生成桩(stub)来模拟协作场景中服务构件的非功能行为，由测试者组合待评估的服务和桩，对评估中的服务进行负载测试。该工具的不足是需要手工编码服务组合。

3.2.2 健壮性测试

面对无效、不正确的输入，Web 服务面临的挑战是要确保其能够正确地处理。Martin 等人[10]通过自动产生和执行 Web 服务请求，分析请求响应对，实现一个健壮性的 Web 测试服务框架 WebSob。具体的过程是：给定提供者的 WSDL 规约，自动生成请求者的 Java 代码，然后利用单元测试自动生成工具产生单元测试，并执行单元测试，最后分析来自服务调用的大量的请求响应对，从中识别潜在的健壮性问题。该方法从服务请求者角度引导测试过程，不需访问服务实现，但手工分类响应类型不能实际反映提供者的响应活动。

文献[11]利用完备性和一致性分析、模式识别技术，提出基于拓扑的正面和负面测试 Web 服务，以保证其可信性，从 OWL-S 规约中产生测试用例。该方法验证了 Web 服务的功能属性和健壮性，并提高了 Web 服务的可扩展性。

3.3 典型的离线测试方法比较

表 1 从方法的提出者、服务规约、测试属性、采用的测试技术、是否有工具支持，以及方法的优缺点对典型的离线测试方法进行比较，总结了当前较为有效的测试框架和工具。需要注意的是，对于有些离线测试服务的测试结果，在线审计和在线监控阶段仍可以重用。

表 1 典型的离线测试方法比较

提出者	服务规约	测试属性	测试技术	工具支持	方法评价
Jiang 等[4]	WSDL	测试定制在合约中服务提供者和使用者的责任	合约变异	WST-DGT 原型	优点：自动生成测试用例；缺点：缺乏理论证明合约变异算子的完备性，不能处理复杂的数据类型
Frantzen 等[5]	WSDL	检验服务的交互行为是否正确	基于 STS	无	优点：能够检验同步和异步调用；缺点：未考虑操作之间的相关性
Wang 等[7]	无	验证模型通用性、负载逼真度、框架可扩展性	分布式实验	Weevil 框架	优点：自动配置实验参数；缺点：未考虑动态负载
Grundy 等[8]	BPMN 或 ViTABaL-WS	评估性能(吞吐量和服务组合的响应时间)	性能测试床生成技术	扩展的 MaramaMTE	优点：自动生成测试床；缺点：只支持静态的服务组合的性能测试
Martin 等[10]	WSDL	分析请求响应对	单元测试	WebSob	优点：能够识别潜在的健壮性问题；缺点：需手工分类响应类型

4 在线审计测试

我们期望在 Web 服务设计和开发早期就能发现错误，大大减少生命周期后期带来的测试开销。然而开发者并不能预期在预发布阶段服务实际的运行环境，所以需要在实际的注册环境中，由部署者在在线审计阶段引入新的策略和方法，测试请求注册的服务。

在线审计阶段主要关注的是服务互操作问题。当待注册的服务被调用时，确定它的行为与规约是否一致，并且检测它是否能够正确地调用已在注册中心发布的服务，以便注册在同一 UDDI 注册中心的服务之间的协作。

4.1 增强的 UDDI 注册中心

Tsai 等人[12,13]利用检入(check-in)和检出(check-out)策略增强 UDDI 注册中心的测试功能。对注册前的服务实施测试，开发了一个分布式 Web 服务测试框架 WSTF，其由 test masters，test agents 和 test monitors 组成，在不同的站点上测试分布的 Web 服务。该方法的主要思想是在服务注册前，服

务提供者提供服务及其相应的测试脚本，测试脚本也可以被用户使用。在注册中心接受一个新的服务之前，test masters 在数据库中查找其相应的测试脚本，然后发送测试命令给 test agents 进行测试。test masters 接收来自 test agents 的测试结果，并报告给 UDDI 注册中心。当测试通过时，UDDI 注册中心执行检入策略，将服务存储在注册中心中。在使用一个服务前，用户检索其测试脚本，对服务进行测试。test monitors 收集服务和用户之间交换的消息，并记录状态的改变，将获取的这些消息发给 test masters 和 test agents。只有当测试通过时，UDDI 注册中心才执行检出策略，将服务提供给用户，这时用户才能使用服务。在该框架中，测试脚本会随着 Web 服务的演化而演化。该框架使用免费的计算资源来执行 Web 服务测试，以减轻 UDDI 注册中心的负担。

4.2 基于 GT 规则的测试

服务描述和发现通常只规定了服务的语法方面，缺少行为语义信息，这需要添加服务的行为规约。Heckel 等人[14]用 GT(Graph Transformation)规则表示服务操作的行为规约(包括单个操作和操作序列)，为单个操作生成测试用例，并基于数据流分析定义两个操作间的依赖关系，由此得到操作序列的测试用例，在服务注册前，由“发现服务”自动测试 Web 服务，来确认实现和规约的一致性。首先“发现服务”从已提供的 WSDL 规约和行为规约中生成测试用例，然后在目标 Web 服务上运行测试用例，如果测试通过，服务则成功发布到注册中心。反之就会产生一份报告，发给服务提供者，提供者根据这份报告提出的问题来修改模型的规则或服务实现。另外，该方法还考虑了服务请求者和提供者之间绑定的可靠性，用户可以使用一个高质量的服务发现 agency 来保证通过测试的服务是可靠的。该方法划分输入子集需依赖于测试者的经验。

4.3 基于 PSM 的测试

在线测试预发布的服务，不仅需要提供服务规约，还需要知道服务之间是如何调用的。Bertolini 等人[15]用 PSM (UML 2.0 Protocol State Machine，协议状态机)来表示服务的交互行为，并且把 PSM 作为参考模型用于生成测试用例。该方法关注的是待注册服务与已注册服务的互操作性。其基本思想是将 PSM 规约转换为符号转换系统 STS(Symbolic Transition Systems)模型，由 STS 的轨迹定义实现与 STS 模型的一致性关系，然后根据 STS 模型和一致性关系自动生成测试用例，并以 on-the-fly 方式实施测试，避免了状态空间爆炸。测试会给 UDDI 注册中心带来一定的开销，但是保证了发布在同一注册中心的服务之间正确的互操作。

与前两个方法有很大不同：前两个方法评估待注册服务的输入输出行为，而 Bertolini 提出的方法是通过服务正确地调用序列，验证待注册服务是否能与已注册服务正确地交互。

4.4 语义 Web 服务的功能一致性测试

如果缺少语义信息，就不能正确地发现和调用服务。Paradkar 等人[16]利用 IOPEs(Inputs，Outputs，Preconditions and Effects，输入、输出、前置条件和结果)范型定义 Web 服务的语义，并提出语义 Web 服务的功能一致性测试。对于每个 Web 服务，其测试目标是由一组错误模型精化的前置条件。一个规划器构件接受这些目标，产生满足测试目标的操作序列，由此得到可执行的测试用例，最后通过接口访问服务实施测试。该方法考虑了两种接口：一种是可以直接调用 Web 服务，另外一种是通过图形用户接口调用服务。该方法的特点是产生的验证序列用来保证由一个结果引起的改变能够正确地被实现。

4.5 典型的在线审计测试方法比较

表 2 给出了典型的在线审计阶段采用的测试方法的比较。需要指出的是，审计阶段可以重用离线阶段采用的基于模型的方法来测试功能属性。

表 2 典型的在线审计测试方法比较

提出者	服务规约	测试属性	属性规约语言	测试技术	工具支持	方法评价
Tsai 等[12,13]	WSDL	功能属性、健壮性、性能和可扩展性	无	检入和检出测试策略	WSTF 框架	优点：测试脚本会随着 Web 服务演化而演化；缺点：未评估测试用例的充分性
Heckel 等[14]	WSDL	服务的行为(单个操作和操作序列)	GT 规则	等价类划分	原型	优点：自动生成测试用例和执行测试过程；缺点：划分输入子集依赖于测试者的经验
Bertolini 等[15]	WSDL	待注册服务与已注册服务的之间互操作性	Protocol State Machine	基于模型的一致性测试	Audition 框架	优点：避免状态空间爆炸；缺点：存在不确定性
Paradkar 等[16]	WSDL	操作序列	IOPEs	边界值分析、变异分析	原型	优点：自动生成测试用例；缺点：需进一步考虑如何以自适应的方式生成测试用例

5 在线监控

在线监控是在实际的运行环境中观察 Web 服务的行为是否与用户期望的行为一致。组合服务中的服务构件是由不同组织提供和控制的，在运行时可能发生改变，而集成者可能不知道这些改变。传统的监控方法是在部署前完成的，由于运行环境不断地变化，或服务本身发生改变，传统的测试方法已不再适用于这种情形，因此需要在运行时动态地持续监控 Web 服务。

为了充分地理解监控方法，我们需要从不同的方面对监控方法进行比较，参考了文献[17]中的比较参数。除了第 2 节已提到的协作模型和待监控属性(功能和非功能属性)，还包括监控规约语言和监控方式。

监控规约语言：规定了问题领域的需求和与系统行为相关的属性。监控器观察系统的行为与给定的属性规约是否一致。

监控方式分为 3 种：①被动监控(passive monitoring)。只观察系统运行时行为，当检测到错误时，不做任何操作；②反应监控(reactive monitoring)。根据被监控行为的分析结果，当错误发生时，人工干预系统行为；③主动监控(proactive monitoring)。预测可能发生的错误，并采取措施避免错误发生。后两种监控方式对系统性能有影响，并对可控制性提出了挑战。

5.1 需求监控

运行时确认软件系统是否满足需求是一个重要问题，这是因为系统在部署前满足需求，然而在运行时可能会违反需求，尤其是对 Web 服务更是如此，运行环境不可预期地改变和不能预见服务交互的行为。当服务动态地组合时，监控运行时服务行为是否与需求一致。

Mahbub 和 Spanoudakis[18,19] 认为运行时监控服务组合应主要考虑行为属性和行为假设这两类需求。从 BPEL4WS 规约中自动获取行为属性，而行为假设需要手工规定，用事件演算(event calculus)表示这些需求。事件演算作为需求的表示语言具有良好的语义特征，能够①规约时态约束；②基于一阶逻辑的推理规则进行推理。运行时收集的事件存储在时态演绎数据库中，用完整性检查技术验证收集的事件是否满足待监控的需求。已开发的原型能够监控 3 种与需求不一致的行为：已记录行为、期望行为和未判别行为。由于收集事件与业务过程是并发执行，因此这种方法对系统性能影响较小，但该方法不能诊断错误。

5.2 自动监控

为了减轻设计和实现监控器的工作，Barbon 等人[20] 提出从规约中自动生成实例监控器和类监控器，监控单个 BPEL 过程实例的行为和实例类的行为。用规约语言 RTML(Run-Time Monitor specification Language)表示待监控的属性，能够检测时态、布尔、与时间相关的统计属性。该方法只能发现错误，属于被动监控。该方法对当前的 BPEL 执行引擎 ActiveBPEL[21] 进行扩展，实现了相应的监控功能。监控功能与业务逻辑是完全分离的，监控过程与业务过程并发执行，不会影响系统性能，但需实验评估监控器的可用性和有效性。

5.3 规划监控

软件系统的规模和复杂性不断增大，很难正确地发现和调用服务。在 SOA 中，业务过程是由多个参与者拥有的，业务过程的改变要被所有的参与者验证，所以要满足所有参与者的业务目标，并且要保证运行时的正确性，获得这样一致的业务过程是很困难的。

Lazovik 等人[22] 针对编舞提出一种规划框架，用高级语言 XSRL(Xml Service Request Language)表示服务请求。由用户发出一个请求，规划器就会综合出一个可执行的规划，并实施监控。如果规划违反了用户的需求，那么就会重新开始一个规划任务。该框架是反应监控，设计者定义 3 种断言：①简单断言。在转换到下一个状态之前，目标必须为真；②保持断言。在整个过程执行中，目标必须为真；③业务实体断言。对某个过程执行的演化序列来讲，目标必须为真。在该框架中规划步骤和执行步骤连续地交织在一起，由于 BPEL 缺乏语义信息，当从 WS-CDL 规约中得到状态转换系统时，使用领域操作符和构造子来丰富其语义信息，能够处理环境的不确定性。然而该方法还存在一些问题，比如如何选取执行路径的优化准则。

5.4 动态监控

当监控过程与业务过程交织在一起时，执行监控功能会使得系统性能降低。针对这样的问题，Baresi 等人[23,24] 提出动态地调整监控策略，从而降低对系统性能的影响。监控器是用 WS-BPEL 实现的服务，能够在标准的 BPEL 引擎上运行，它作为外部服务可被业务过程调用。由 WS-CoL(Web Service Constraint Language)规约语言定义监控规则来标注 BPEL 过程，监控器用来分析 WS-CoL 约束。该方法在设计规约阶段时，业务逻辑与监控功能是分离的，但当需要执行监控功能时，允许业务过程与外部的监控器交互，采用面向方面的方法将监控器的 BPEL 代码添加到业务过程中。在运行时阶段，阻塞业务过程执行，检查前置和后置条件来及时发现错误，业务过程与监控过程是未分离的。当检测到错误时，就会触发相应的恢复活动。该方法收集数据的来源有两个：直接从过程获取或从外部数据源获取。该方法实现了一个原型工具 Dynamo，能够监控功能属性和超时、运行时错误。

由于运行时执行环境动态地变化，需要服务自适应于这种变化。文献[25]通过采用面向方面的方法拦截 SOAP 消息，实现了 VieDAME 系统。根据各种 QoS 属性以非纠缠的方式监控 BPEL 过程，并基于各种置换策略来置换已有的服务。所选择的置换服务在语法和语义上等价于在 BPEL 过程中使用的接口。该系统具有良好的可扩展性，在运行时服务能够自动地被置换，而不会影响系统性能。

5.5 SLA 监控

在运行时持续监控相关的质量参数，需要精确地检测到任何 SLA 违例。Raimondi 等人[26] 用 SLA 表示延迟、吞吐量、可用性和可靠性，再将 SLA 转换成时间自动机(timed automata)，实现了一个基于时间自动机的监控器。为 SOAP 消息添加时间标签作为 timed letters，将检测 SLA 违例问题归结为时间自动机是否可以接收时间字(timed words)，这可以在多项式时间内处理完成。该方法只考虑了服务的时限性问题，还需考虑 SLA 可能会强加非时态属性的需求。

在移动环境中资源受限的情况，由于用户存储和带宽有限，不能为 SLA 检验提供有价值的资源，而且会由于加载 SLA 管理引发不必要的违例，这给扩展 SLA 监控各种环境提出了难以解决的问题。Bertolino 等人[27] 提出自适应的 SLA 检验策略：计算某个用户行为与违例的距离，如果用户行为接近于违例，那么要频繁检验该用户行为，以增加检测到违例的可能；如果在某个时间段减少对用户行为的检测，那么说明该用户对协议违例的可能性较小，从而自适应地调整对某个用户的检验频率，使得可用的资源提供给用户。该方法优化了资源使用，具有能够很好适应 Web 服务不稳定情况的特点。

SLA 涉及服务提供者和用户，如果运行时监控忽视了服务参与者中的一方，那么 SLA 中定义的责任和权利都是无用的。为了确保每个参与者与其应履行的责任一致，需要参与者双方互相监控。文献[28]提出可监控性概念，是指参与者双方能够互相观察到与 SLA 相关的服务行为，或是由参与者信任的第三方来观察参与者双方的行为，并提供模型和分析技术来推理 SLA 的可监控性。

5.6 Policy 监控

Erradi 等人[29] 开发了基于策略的 MASC(Manageable and Adaptive Service Compositions)中间件，用于监控组合服务的业务异常和运行时错误，并能够自适应各种改变按需提供的监控功能。通过定义新的监控类型和控制策略断言来扩展 WS-Policy，由此得到 WS-Policy4MASC 语言来规定可监控的需求。MASC 中间件提供了 SOAP 消息层和业务编排层同步和异步监控，及时地触发相应的监控服务。自适应的

监控策略使得监控开销较小。另外，业务过程与监控服务的分离使得 MASC 中间件具有良好的可扩展性和可重用性。

5.7 工业界方法

近年来，工业界已开发了许多监控工具。与研究领域的方法比较，业界工具更多是从服务提供者的角度，监控低级别的事件和某些非功能属性，并且监控功能只是部署环境的一部分，其中具有代表性的方法有 Cremona 和 Colombo。

(1)Cremona

IBM 开发了 Cremona(Creation and Monitoring of Agreements)框架[30]，确立了用户和提供者之间的协商，并在生命周期管理 WS-Agreement，如创建、中止、运行时监控和再协商。WS-Agreement 定义了接口、安全性和 QoS 属性，从而保证提供者提供服务的功能或非功能属性的正确性，并保证满足用户的需求。

Cremona 框架简化了协定(Agreement)的定义，运行时监控协定的状态。该框架提供了一个协定提供器构件，根据提供的一个协定，检查系统可提供的资源，确定用户和提供者之间的协定是否被接受。如果用户和提供者都接受了这个协定，那么就会在运行时由一致性监控器检测该协定的有效性。该框架不仅能够检测到违例，并能预测将要发生的违例，采取正确的措施。

(2)Colombo

Colombo[31]是一个面向服务体系结构的轻量级中间件，也是由 IBM 开发的，提供优化的运行时环境，并为开发和部署提供简化模型，具有更好的运行时性能。Colombo 支持 BPEL 服务组合和 WS-Policy。WS-Policy 是一个声明性语言，用来定义 BPEL 过程的 QoS 属性，如安全性、事务和可靠的消息。该工具能够及时地发现错误行为，但监控过程与业务过程没有分离，执行监控功能会对系统性能有所影响。

5.8 典型的监控方法的比较

表 3 给出了典型的在线监控方法的比较。需要说明的是，虽然监控服务会检测到运行时错误，但会引入诸多问题，如收集运行时数据会使系统性能降低，或是增加系统开销，所以研究者要保证观察 Web 服务行为的执行不会影响系统性能。另外，大多数方法都处于被动和反应监控，如何主动监控运行时错误仍然是一个热点问题。

表 3 典型的在线监控方法比较

提出者	协作模型	待监控属性	监控规约语言	监控方式	确认技术	工具支持	方法评价
Mahbub 等[18,19]	基于 BPEL 的编排	行为属性和行为假设	事件演算	被动监控	基于时态演绎数据库的完整性检查技术	原型	优点：采用的监控语言具有良好的语义；缺点：不能诊断错误
Barbon 等[20]	基于 BPEL 编排	时态、布尔、统计和与时间相关的属性	RTML	被动监控	具体的实现技术	扩展 ActiveBPEL 引擎	优点：监控器与业务过程完全分离；缺点：需实验评估监控器的可用性和有效性
Lazovik 等[22]	基于 WS-CDL 编舞	服务请求	XSRL	反应监控	断言检测	XSRL 框架	优点：解决环境的不确定性；缺点：需优化执行路径
Baresi 等[23,24]	基于 BPEL 编排	与外部服务的交互行为	WS-CoL	反应监控	断言检测	Dynamo	优点：处理同步问题。缺点：监控过程影响系统性能
Raimondi 等[26]	基于 SOAP 消息的交互	延迟、可靠性和吞吐量	SLA	反应监控	基于时间自动机的轨迹分析技术	基于时间自动机的监控器	优点：接收检测器性能与系统中的消息总数无关；缺点：未考虑 SLA 中非时态属性的需求
Erradi 等[29]	基于 BPEL 的编排	业务异常和运行时错误	WS-Policy4MASC	反应监控	策略监控	MASC 中间件	优点：能够自适应各种改变按需提供监控功能；缺点：需考虑集成已有的标准，如 WS-Agreement
Ludwig 等[30]	协定发起者和服务提供者之间的交互	接口、安全性和 QoS 属性	WS-Agreement	主动监控	创建协定并监控其状态	Cremona	优点：能够预测将要发生的违例，并采取正确的措施；缺点：WS-Agreement 是静态的，不易扩展，且不能够描述协定间的关系
Curbera 等[31]	基于 BPEL 编排	安全性、事务和可靠的消息	WS-Policy	被动监控	策略监控	Colombo	优点：提供优化的运行时环境；缺点：执行监控功能会对系统性能有所影响

结束语 本文从 Web 服务生命周期的角度分析和总结了 Web 服务测试方法和技术的现状，但还需要从以下几个方面进行深入探讨：

(1) 由于服务构建系统的复杂性不断增加，因此需要测试工具具有某种程度的自动化，如自动生成测试输入，或自动化测试过程，或提供集成的测试环境能够自动地产生支撑代码。另外，用户动态地绑定服务需进行重测试，除了第三方测试，用户没有测试服务的经验和领域知识，这也需要自动化测试工具来引导测试过程，反复地、快速地实施测试。

(2) 测试评估 QoS 属性与功能属性同等重要，涉及到有关 QoS 参数配置和运行环境。评估性能(如服务的响应时间)和可靠性，由于测试这些属性不仅取决于测试中的服务，而且还与支撑系统和网络有关，这就引入了一些问题：如何使得服务与支撑平台完全分离，来评估给定的 QoS 属性？

(3) 在审计阶段需要评估服务的上下文感知和自适应性，尽管运行时监控提供了有关服务自适应性的行为证据，但是错误行为造成的结果在运行时阶段很难移除，另外在离线阶段处理这些问题(如开发仿真平台)所需开销太大，所以需要在审计阶段引入评估服务的上下文感知和自适应性的策略。

参 考 文 献

[1] OASIS. Web Service Business Process Execution Language Version 2.0 Specification[S]. OASIS standard, April 2007

[2] Kavantzas N, Burdett D, Ritzinger G, et al. The web service choreography description language v1.0. W3c recommendation, W3C, Nov. 2005

[3] Martin D, Burstein M. OWL-S: Semantic markup for web services[D]. DAML Services, November 2004. http://www.daml.org/services/owl-s/1.0/

(下转第 62 页)

[J]. 计算机研究与发展,2008,45(1):83-89

[21] Handy M J, Haase M, Timmermann D. Low energy adaptive clustering hierarchy with deterministic cluster-head selection[C] //Proc. of the 4th IEEE Conf. on Mobile and Wireless Communications Networks. Stockholm: IEEE Communications Society, 2002:368-372

[22] 肖琳,程利娟,王福豹. 一种低功耗无线传感器网络时间同步算法[J]. 计算机研究与发展,2008,45(1):126-130

[23] Dong S K, Seung Y L, Kwang H W, et al. Time-synchronized forwarding protocol for remote control of home appliances based on wireless sensor Network[J]. IEEE Trans. on Consumer Electronics, 2007, 53(4):1427-1433

(上接第 34 页)

[4] 姜瑛,辛国茂,单锦辉,等. 一种 Web 服务的测试数据自动生成方法[J]. 计算机学报,2005,28(4):568 - 577

[5] Frantzen L, Tretmans J, Vries R D. Towards Model-based Testing of Web Services [C] // International Workshop on Web Services - Modeling and Testing - WS-MaTe. 2006:67-82

[6] Keum C, Kang S, Ko I Y, et al. Generating test cases for web services using extended finite state machine[C] // Proc. of IFIP TestCom. 2006:103-117

[7] Wang Y Y, Rutherford M J, Carzaniga A, et al. Automating Experimentation on Distributed Testbeds[C] // Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE 2005). 2005

[8] Grundy J, Hosking J, Li L, et al. Performance Engineering of Service Compositions [C] // Proc. of the 2006 International Workshop on Service-oriented Software Engineering. ACM Press, 2006:26-32

[9] Bertolino A, Angelis G D, Polini A. A QoS Test-bed Generator for Web Services[C] // Proc. of ICWE 2007. Springer, 2007

[10] Martin E, Basu S, Xie T. Automated Testing and Response Analysis of Web Services[C] // 2007 IEEE International Conference on Web Services (ICWS 2007). 2007

[11] Tsai W T, Wei X, Chen Y. A Robust Testing Framework for Verifying Web Services by Completeness and Consistency Analysis[C] // IEEE International Workshop on Service-Oriented System Engineering (SOSE). Beijing, October 2005:151-158

[12] Tsai W T, Paul R, Cao Z, et al. Verification of Web Services Using an Enhanced UDDI Server[C] // Proc. of IEEE WORDS. 2003:131-138

[13] 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

[14] Heckel R, Mariani L. Automatic conformance testing of web services[C] // Proc. Fundamental Approaches to Software Engineering (FASE 05). 2005:34 -48

[15] Bertolino A, Frantzen L, Polini A, et al. Audition of web services for testing conformance to open specified protocols[C] // Architecting Systems with Trustworthy Components. Springer-Verlag, 2006

[16] Paradkar A M, Sinha A, Williams C, et al. Automated Functional Conformance Test Generation for Semantic Web Services[C] // IEEE International Conference on Web Services (ICWS 2007). 2007:110-117

[17] Delgado N, Gates A, Roach S. A taxonomy and catalog of runtime software-fault monitoring tools [J]. IEEE Trans. Software Enginerrning, 2004, 30(12):859-872

[18] Mahbub K, Spanoudakis G. A framework for requirements monitoring of service-based systems[C] // Proceedings of the 2nd International Conference on Service Oriented Computing. 2004: 84-93

[19] Spanoudakis G, Mahbub K. Non Intrusive Monitoring of Service Based Systems[J]. International Journal of Cooperative Information Systems, 2006, 15(3):325-358

[20] Barbon F, Traverso P, Pistore M, et al. Run-time monitoring of instances and classes of Web service compositions[C] // IEEE International Conference on Web Services (ICWS'06). 2006:63-71

[21] ActiveBPEL. The Open Source BPEL Engine. http://www.activebpel.org

[22] Lazovik A, Aiello M, Papazoglou M. Associating assertions with business processes and monitoring their execution[C] // Proceedings of the Conference on Service-oriented Computing (ICSOC-04). ACM Press, 2004:94-104

[23] Baresi L, Ghezzi C, Guinea S. Smart monitors for composed services[C] // Int. Conf. on Service-oriented Computing (ICSOC-04). ACM Press, Nov. 2004:193-202

[24] Baresi L, Guinea S. Towards dynamic monitoring of WS-BPEL processes[C] // Proc. of the 3nd International Conference on Service-oriented Computing (ICSOC'05). 2005:269-282

[25] Moser O, Rosenberg F, Dustdar S. Non-intrusive monitoring and service adaptation for WS-BPEL[C] // Proceeding of the 17th International Conference on World Wide Web. 2008:815-824

[26] Raimondi F, Skene J, Emmerich W. Efficient online monitoring of Web-service SLAs[C] // Proceedings of ACM SIGSOFT/FSE 2008. Atlanta, USA, 2008

[27] Bertolino A, Angelis G D, Elbaum S, et al. Scaling up SLA Monitoring in Pervasive Environment [C] // Proceedings of Int. Workshop on the Engineering of Software Services for Pervasive Environment (ESSPE 2007) at ESEC/FSE. 2007

[28] Skene J, Skene A, Crampton J, et al. The monitorability of service-level agreements for application-service provision[C] // Proceedings of WOSP 2007. Buenos Aires, 2007:3-14

[29] Erradi A, Maheshwari P, Tosic V. WS-Policy based Monitoring of Composite Web Services[C] // Fifth European Conference on Web Services (ECOWS'07). 2007:99-108

[30] Ludwig H, Dan A, Kearney R. Cremona: An architecture and library for creation and monitoring of WS-agreements[C] // Proceedings of the 2nd International Conference on Service-oriented Computing(ICSOC 2004). New York, USA, ACM, 2004:65-74

[31] Curbera F, Duftler M J, Khalaf R, et al. Colombo: lightweight middleware for service-oriented computing [J]. IBM Syst. J, 2005, 44(4):799-820