Web服务自动化测试技术

马春燕 朱怡安 陆 伟
(西北工业大学软件与微电子学院 西安 710072)

摘 要 Web服务(Web services)已成为当前和未来网络分布式应用的主流软件开发技术。如何确保Web服务软件的质量和可靠性是当前软件工程领域关注的焦点问题。分析了Web服务测试的层次和阶段,以及测试工具的现状,提出了Web服务自动化测试的技术框架,分析了此框架内Web服务操作、Web服务操作序列和Web服务组合WSBPEL流程测试的关键技术,并研制了测试用例自动生成的原型系统,给出了实验结果,最后指出Web服务自动化测试技术值得进一步探讨的主题。

关键词 Web服务,自动化测试技术,测试用例自动生成

Automatic Test for Web Services

MA Chun-yan ZHU Yi-an LU Wei
(Department of Software and Microelectronics, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Web services has become the main software technology in the distributed application currently and in the future. How to assure the quality and dependability of the Web services is the focus in the current software engineering. Because of the technology complexity, network and distributed deployment, and dynamic application connections of the Web services, traditional software testing technique can't fully meet the testing requirements of the Web services. The paper analyzed the level and stage of Web services testing, current testing tools. Based on the outcome of our team, the paper proposed automated testing framework and the key issues, developed the prototype systems for automatic test case generation, and gave the experimental results. At last, Web services testing techniques which should be further explored were expressed.

Keywords Web services, Automatic test, Automatic test case generation

1 引言

Web服务(Web services)是针对因特网上分布计算提出的一种基于开放标准、松散耦合及跨平台的新型软件构件,已广泛应用于通信、金融、地理信息、电子商务、电子政务、手持设备等领域,成为当前和未来网络分布式应用的主流软件开发技术。如何确保Web服务软件的质量和可靠性是当前软件工程领域关注的焦点问题。由于Web服务技术规范的复杂性、应用部署的网络分布性和在线运行形态的多变瞬时性等特点,传统的软件测试技术已不能完全满足其测试工作的要求。除此之外,Web服务提供了一种机器对机器的通讯机制,其软件测试策略主要基于服务接口进行设计和实现,与传统需要大量人工干预的测试方法不同,难以通过手工测试方式完成,需采用自动化测试方法。因此,对传统软件测试技术和方法进行分析研究和扩展,提出针对Web服务软件的新的测试技术和方法,开发相应的自动化测试支持工具,具有十分重要的理论意义和实用价值。

本文对Web服务软件测试用例的自动生成技术进行了深入研究;第2节分析了Web服务测试层次和测试阶段的基础上,第3节提出了基于模型的Web服务自动化测试的技术框架,并研究了在此框架内需解决的关键技术;第4节根据作者研究团队的成果,设计并实现了Web服务测试用例自动生成的原型系统,并采用原型工具对实际案例进行了应用研究;第5节详细讨论了Web服务自动化测试技术值得进一步研究的主题。

2 Web服务测试分析

2.1 Web服务测试工具现状

1) WebInject

WebInject[1]是一个执行Web服务和产生测试结果报告的引擎,该工具需要读入一个包含描述测试用例的XML文件,测试用例中包含需要测试的操作的描述、测试数据以及需要测试的Web服务的URL。

2) WStest

用户通过WStest[2]输入需要测试的Web服务的地址、要测试的操作名以及激活相应操作的测试用例。该工具自动激活用户指定的Web服务,并报告测试返回的结果,用户通过查看返回的结果与预期结果是否相符来判断该服务中此操

到稿日期:2011-03-31 返修日期:2011-06-24 本文受陕西省自然科学基础研究计划项目(2009JM8003-5)资助。
马春燕(1978—),女,博士,讲师,主要研究方向为软件测试、任务关键软件的建模和仿真,E-mail:machunyan@nwpu.edu.cn;朱怡安(1961—),男,教授,博士生导师,主要研究方向为软件工程;陆 伟(1975—),男,博士,讲师,主要研究方向为自愈技术。

作功能的正确性。

3)SoapUI

SoapUI[3]把一个或多个测试套件(TestSuite)组织成项目,每个测试套件包含一个或多个测试用例(TestCase),每个测试用例包含一个或多个测试步骤,诸如发送请求、接受响应、分析结果、改变测试执行流程等。

4)TestMaker

TestMaker[4]的测试是通过"测试代理(test agents)"的脚本来完成的。TestMaker 提供一个"代理向导"(agent wizard)读入 WSDL 并自动创建一个测试代理的基本结构,测试人员需要检查测试代理的源代码填写缺少的内容。

5)WebServiceTester

WebServiceTester[5]提供 Web 服务自动化测试的产生、功能测试、回归测试、负载测试,安全测试和诊断及 BPEL 测试。

6)QEngine

QEngine[6]自动进行负载测试,用户可以通过该工具从 WSDL 文档生成测试脚本,并审核实际的响应结果。

现有的 Web 服务测试工具都要求用户手工设计、产生和撰写测试用例,而测试用例的自动生成是 Web 服务测试自动化的关键所在。本文在研究 Web 服务测试相关理论和技术的基础上,提出 Web 服务测试用例自动生成技术框架,并根据研究团队的研究成果,开发相应的测试用例生成支持工具。

2.2 Web 服务测试的层次和阶段

文献[7-9]等把 Web 服务分为无状态 Web 服务和状态 Web 服务两类,定义如下。

定义 1(无状态操作) 如果对一个操作指定同样显式的输入参数,该操作产生同样的执行结果,则称该操作是无状态的。

定义 2(状态操作) 如果对一个操作指定同样显式的输入参数,该操作可能有不同的执行结果,则称该操作是有状态的。

定义 3(无状态 Web 服务和状态 Web 服务) 如果 Web 服务所有操作都是无状态的,则称该 Web 服务为无状态 Web 服务,否则称其为状态 Web 服务。

无状态 Web 服务的测试仅需要独立测试其各个操作即可;而状态 Web 服务操作序列的测试更为重要,因为不同的操作序列常常导致 Web 服务进入不同的数据状态。状态 Web 服务操作执行的结果取决于 Web 服务已经执行过的操作及其被调用的次序,亦即取决于 Web 服务所收到的消息序列。因此,对于状态 Web 服务,操作序列的测试更为重要,它可验证状态 Web 服务的行为。

如图 1 所示,论文将 Web 服务测试分为 3 个层次,即 Web 服务单个操作的测试(即无状态 Web 服务的测试)、Web 服务操作序列的测试(即状态 Web 服务的测试)以及 Web 服务组合测试。前两个层次均为单独的 Web 服务测试,对应于传统软件的单元测试,Web 服务组合测试对应于传统软件的集成测试;根据 Web 服务测试的层次和软件测试阶段[10-12]的划分,论文将 Web 服务的测试过程划分为 4 个阶段,即软件行为建模(或仅仅是模拟单个操作的输入)、测试用例的生成、测试的执行和评估以及测试过程的度量。

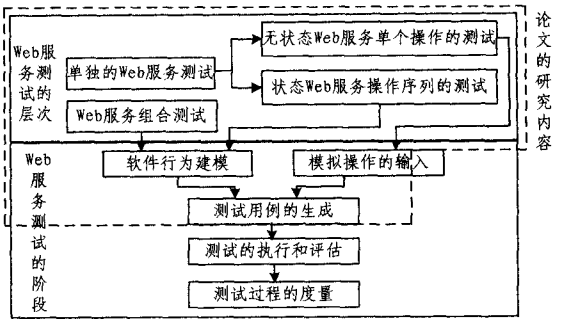

图 1 Web 服务测试的层次和阶段

目前存在的 Web 服务测试工具的主要功能集中在 Web 服务测试过程的第三阶段和第四阶段,对测试用例的执行和评估结果的过程等具有辅助作用。软件测试的本质是针对测试对象确定一组测试用例,所以,测试用例在软件测试中占据中心地位。测试用例的生成在整个测试活动中位于第二阶段,而如何构造测试用例则取决于第一阶段程序行为的建模(或模拟操作输入的手段)。论文重点关注 Web 服务软件测试过程的第一阶段和第二阶段,即图 1 中虚线标示的矩形框。下面重点对软件测试过程的第一阶段和第二阶段的相关研究内容进行论述。

1)软件行为建模或模拟操作的输入

模拟软件和它的运行环境之间的交互是测试者的首要任务。测试者必须识别和模拟软件系统的接口,这涉及到列举接口所有的输入和输出,构建所有可能的输入序列,所以测试者需要对软件的行为进行建模。测试者可以构建一个行为模型来模拟一个软件是如何操作的,再基于该行为模型,产生测试用例。通常,该行为模型也称为测试模型。例如,在白盒测试中,常用的测试模型包括控制流图、数据流图和程序输入/输出依赖图等;在黑盒测试中,有限状态机及扩展有限状态机、petri 网和马尔科夫链等都可以作为软件的测试模型。软件行为建模通常包括测试模型选取和测试模型构建两个步骤。

(1)测试模型选取

如何选取具体测试领域的测试模型可以参考文献[13]描述的 3 个因素,即衡量软件行为或结构的测试模型是否适用于具体的应用领域涉及以下 3 项内容:

a)测试模型开发的难易程度和自动化程度;

b)测试模型是否包含测试用例生成所需的全部信息;

c)从测试模型获取测试用例的难易程度。

若选取的测试模型适用于具体的应用领域,则测试模型易于开发,可以半自动化或自动化构造;测试模型包含生成测试用例所需的全部信息;基于测试模型获取测试用例较容易。

(2)测试模型构建

测试模型构建是任何测试过程中最基本的阶段,是其它 3 个测试阶段的基础,也是自动化测试的关键步骤。白盒测试方法和黑盒测试方法都要构造测试模型,但是通常文献中描述的基于模型的软件测试一般属于黑盒测试的范畴[14],本文对此没有进行刻意区分。在白盒测试方法中,测试模型(如控制流图和数据流图等)的构建往往作为测试工作的重点;但是在黑盒测试方法中,人们却很少关注测试模型的构建。测试模型的构造是基于相应测试模型的测试技术得以广泛应用的基础,若没有系统的指导方法,测试模型的手工构造过程将

会非常复杂，错误的构造可能导致测试模型与软件的规格说明或源码结构不一致。

2)测试用例生成

软件测试的本质是针对测试内容确定一组测试用例，测试用例是测试工作的指导，也是评估测试结果的度量基准。文献[15]给出了测试用例的定义。

(1)为特定目标而设计的一组测试输入、执行条件和期望的输出结果。

(2)为某一测试项设计的特定输入、预测结果和一组执行条件。

由此可以看出，测试用例主要包括测试输入数据和期望结果。测试用例的自动生成是自动化测试技术的重要研究内容，如果运用手工方式设计测试用例，不但要耗费大量的人力资源，还需要很长的测试周期，测试效率低下，并严重依赖于测试人员的个人经验，即使很简单的程序也可能由于个人思维习惯导致测试的遗漏，而且存在测试用例设计过程不规范的问题；另外，随着软件系统规模和复杂性的急剧增加，手工为其生成测试用例非常困难。

基于阶段一生成的测试模型，可能生成无穷的测试用例，所以为了得到最重要的测试用例，应该指定测试的充分性标准。基于 Stream X-machine 模型测试的研究工作如文献[16-19]，在设计的测试条件下都给出了充分的测试用例集的计算方法。

由于在很多文献和书籍中，测试输入和测试用例混合使用，测试路径又决定了测试输入，因此本文将生成的测试输入和测试路径皆称为测试用例，这并不影响研究成果的展现。

3 Web 服务自动化测试的技术框架

根据 Web 服务测试的层次和阶段以及 Web 服务测试工具的相关现状，论文提出了基于模型的 Web 服务自动化测试的技术框架，如图 2 所示。在该技术框架下，对单独 Web 服务的测试是黑盒测试。由于服务代理者及服务请求者通常只能获得 Web 服务的规格说明文档(WSDL 文档或语义规格说明文档)，因此对 Web 服务只能进行黑盒测试；而对 BPEL 流程的测试是基于 BPEL 流程源码的白盒测试，适用于 Web 服务的集成者对流程的自动化测试。下述内容详细阐述了在图 2 所示的技术框架下作者团队解决的关键技术。

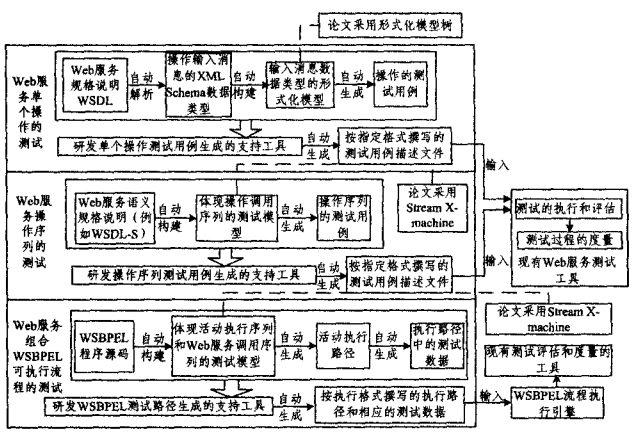

图 2 Web 服务自动化测试的技术架构

3.1 Web 服务操作测试的关键技术

图 2 所示的技术框架采用 WSDL 规格说明测试的单个操作。为了测试 Web 服务的单个操作，测试者需要为操作提供输入消息实例，即为操作提供测试用例，而输入消息的 XML Schema 数据类型决定了相应测试用例的生成。为了基于 XML Schema 数据类型自动生成测试用例，需要直观展现 XML Schema 数据类型的形式化模型，基于该形式化模型自动生成操作的测试用例，在此基础上研发单个操作测试用例自动生成的支持工具，并结合现有的 Web 服务测试工具对 Web 服务单个操作进行自动化测试。

对于 Web 服务单个操作测试用例的自动生成，作者研究团队提出了基于形式化模型树的 Web 服务操作测试用例生成技术[20]，解决了如下关键问题。

1)提出了对输入消息的 XML Schema 数据类型进行建模的形式化模型树，它具有以下优势：

(1)形式化模型树可以通过解析 WSDL 自动构造；

(2)形式化模型树包含单个操作输入消息测试用例产生的所有信息，它可以对 XML Schema 数据类型及其各种约束刻面建模；

(3)可以较容易地从该形式化模型树获取操作输入消息的测试用例。

2)提出了由 WSDL 规格说明自动构造的输入消息数据类型形式化模型树的算法。

3)提出了基于形式化模型树自动生成输入消息测试用例的算法。

3.2 Web 服务操作序列测试的关键技术

广泛使用的 Web 服务描述语言 WSDL 没有提供 Web 服务的行为信息(例如操作的前置条件和结果)，Web 服务代理者和服务请求者仅依据 WSDL 无法完成操作序列的测试。为了克服 WSDL 没有提供行为信息的限制，已经开展了一些研究，例如，W. T. Tsai 等人提出从输入\输出依赖和操作激活序列等 4 个方面扩展 WSDL 以利于对 Web 服务进行测试[21]；Antonia Bertolino[22]等人提出服务提供者注册服务时要提供 WSDL 和建模操作序列的协议状态机；Reiko Heckel[23]等人也提出服务注册时，应上载 WSDL 文档以及 GT 规则(Graph transformation rules)以增加 Web 服务的行为描述，便于测试。但是这些对 WSDL 的扩展由于增加了服务开发的难度和工作量，而且没有相应的规范，因此很难得到服务提供者的广泛支持。

目前，W3C 提出了一些新的提供行为信息的语义 Web 服务标准(例如 WSDL-S[24]，OWL-S[25]等)，并得到了服务提供者的支持。语义 Web 服务为服务发现、激活及组合等提供了很大便利，也提供了更多有利于 Web 服务测试的行为信息。

图 2 所示的技术框架采用语义 Web 服务，自动构建体现操作调用序列的测试模型，自动生成操作序列的测试用例；并在此基础上研发操作序列测试用例自动生成的支持工具，结合现有的 Web 服务测试工具对对状态 Web 服务进行自动化测试。

对于状态 Web 服务操作序列测试用例的自动生成，作者研究团队提出了基于 Stream X-machine 的 Web 服务操作序列测试用例生成技术，解决了以下关键问题。

1)选取测试模型

状态 Web 服务有复杂的持久数据变量，行为状态包括控

制状态(操作序列的体现)和数据状态(持久数据变量值的体现)。因此与传统面向对象软件的行为测试和其他的构件行为测试类似,对状态 Web 服务的测试,可以 EFSM 为测试模型,Stream X-machine(SXM)作为测试模型更优于 EFSM[26];而且 Web 服务每个操作都可以看作一个独立可计算和可测试的单元。所以研究团队提出采用 Stream X-machine 作为状态 Web 服务的测试模型[27],具体优点总结如下:

(1)SXM 可以描述 Web 服务所有可能的操作序列(通过将操作建模为处理函数作为迁移标记来实现);

(2)SXM 可以描述 Web 服务所维护的持久数据变量值的变化(通过内存值的改变展现持久数据变量值的变化);

(3)SXM 可以描述 Web 服务各种可能的使用方式,以检验语义 Web 服务描述和其实现的一致性;

(4)满足一定的测试条件,基于 SXM 测试用例的生成可以自动化完成,而且测试结果的验证也较为容易,即通过检查 SXM 内存的变化和操作的输出结果可以验证执行结果与预期结果是否一致。

2)测试模型的自动构建

SXM 开发的难易和自动化程度是 SXM 作为测试模型对 Web 服务进行自动测试的关键。如果要求状态 Web 服务提供者用 SXM 扩展 WSDL 并进一步提供其行为规格说明,很难得到软件 Web 服务提供者的支持,所以这是基于 SXM 的测试得以广泛应用的障碍。作者的研究团队研究了由 Web 服务提供者广泛支持的语义规格说明 WSDL-S(主要利用操作的前置条件和结果,因为它规定了 Web 服务操作之间的语义依赖,即 Web 服务操作以一个合理的顺序被调用时,状态 Web 服务行为的正确性才有保障)自动构造 SXM 的系统方法[27]。

3)操作序列测试用例的生成

对于由语义服务规格说明构造的测试模型 SXM,需要提出如何满足 SXM 测试条件[28]的解决方案,以使操作测试序列的测试用例可以自动生成。

在前期工作[27]的基础上,对于由语义 Web services 生成的测试模型 DSXM(即确定的 SXM),本文分两个方面讨论如何为确定的 SXM 创造测试条件。

1)创建定义完全的 DSXM

为了创建定义完全的 DSXM,每个状态拒绝的输入可以通过一个错误的转换引向自身,使得所构造的 DSXM 是满足定义完全的。例如,对于文献[27]中的 multimedia conference 的测试模型 DSXM Z′,可以引入 4 个额外的处理函数,代表 Z′拒绝的输入类型导致错误的处理函数,即 $error_{\sigma 1}$,$error_{\sigma 2}$,$error_{\sigma 3}$ 和 $error_{\sigma 4}$。这些处理函数各自的输入分别为 σ1,σ2,σ3 和 σ4,产生了错误的输出,所以,multimedia conference 定义完全的 DSXM Z′除了文献[27]中所示的转换外,还包含如下额外的自循环转换:

(1)从状态 q_0 出发的代表错误行为的自循环转换上的迁移标记有:$error_{\sigma 2}$,$error_{\sigma 3}$,$error_{\sigma 4}$;

(2)从状态 q_1 出发的代表错误行为的自循环转换上的迁移标记有:$error_{\sigma 1}$,$error_{\sigma 4}$;

(3)从状态 q_2 出发的代表错误行为的自循环转换上的迁移标记有:$error_{\sigma 4}$;

(4)从状态 q_3 出发的代表错误行为的自循环转换上的迁移标记有:$error_{\sigma 1}$;

(5)从状态 q_5 出发的代表错误行为的自循环转换上的迁移标记有:$error_{\sigma 1}$;

2)处理函数的集合 Φ 满足输出可区分性

由输出可区分性的定义[29]知,multimedia conference 的处理函数 o_4 和 o_5 及 o_8 和 o_9 不满足输出可区分性。文献[16,29]给出了使得 Φ 满足输出可区分性条件的方法,即使一些内存变量为可观察的,和/或将某些处理函数分割为两个或更多部分。对于 multimedia conference 的提供者而言,他们有 Web 服务的源码,因此完全可以通过观察持久数据变量 participans 的内存值,使上述处理函数满足输出可区分性。对于 multimedia conference 的代理者和请求者而言,内存变量是不可观察的。为了便于 Web 服务代理者和 Web 服务请求者对其进行测试,提出了一个使 Φ 满足输出可区分性的新策略,即将 o_4 和 o_5 以及 o_8 和 o_9 分别封装为 o_4'和o_5'以及o_8'和o_9',使 o_4'和o_5'以及o_8'和o_9'满足输出可区分性,例如:

(1)$o_4'(m,\sigma_3)=(1,m')$,其中,$m'=$ {ConferenceInfo.NumberOfParticipants+1, participants+1}。

(2)$o_5'(m,\sigma_3)=(2,m')$,其中,$m'=$ {ConferenceInfo.NumberOfParticipants−1, participants−1}。

(3)$o_8'(m,\sigma_4)=(3,m')$,其中,$m'=$ { ConferenceV, addMediaForParticipant}。

(4)$o_9'(m,\sigma_4)=(4,m')$,其中,$m'=$ {ConferenceV, deleteMediaForParticipant}。

其中,封装之后的 o_4'实现的功能与 o_4 一致,即 o_4'的执行调用 o_4,并返回一个常量 1 来满足输出可区分性,其余 3 个封装的处理函数与之同理。封装工作可以由 Web 服务代理者和 Web 服请求者完成,他们可以撰写一个本地 Web 服务 multimedia conference′,multimedia conference′的操作(即 o_1',o_2',o_3',o_4',o_5',o_6',o_7',o_8'和o_9')通过调用 multimedia conference 的操作(即 o_1,o_2,o_3,o_4,o_5,o_6,o_7,o_8 和 o_9)来实现其功能。multimedia conference′的操作个数、操作的功能及操作的输入与 multimedia conference 的操作个数、操作的功能及操作的输入保持一致,除了 o_4'、o_5'、o_8'和o_9'与 o_4、o_5、o_8 和 o_9 的输出不一样外,multimedia conference′其它操作的输出与 multimedia conference 相应操作的输出也一样。由此可见,测试 multimedia conference′就相当于测试 multimedia conference。

3.3 Web 服务组合 WSBPEL 流程测试的关键技术

网络上发布的 Web 服务通常结构比较简单、功能单一,它自身可能无法满足企业复杂应用的需要,这就要求将 Web 服务进行组合。为适应 Web 服务组合技术的需求,在过去几年中,出现了 WSBPEL(Web Services Business Process Execution Language)[13]和 WS-CDL(Web Service Choreography Description Language)[14]等基于流程的 Web 服务合成语言。其中,WSBPEL(在本文以后的描述中简称 BPEL)已成为当前进行服务组合的首选语言。

WSBPEL 程序的质量对于企业而言至关重要,因为 WSBPEL 流程出现故障可能会在经济上给企业造成极严重的负面影响。伴随着越来越多的工作流程使用 WSBPEL 建模,为确保高质量的 WSBPEL 代码,WSBPEL 流程的测试亟待解决。对于大型复杂的 WSBPEL 流程,手工产生其测试用

例是繁琐的、困难的，甚至是不可能的，因此如何降低 WSBPEL 程序测试成本，提高测试效率和测试的自动化程度是 WSBPEL 测试中一个亟需研究的问题。

图 2 所示的技术框架基于 WSBPEL 流程源码，自动构建体现活动执行顺序以及 Web 服务激活顺序的测试模型；然后基于该测试模型自动生成流程中活动的执行路径；并为活动执行路径生成相应的测试数据，使得其测试用例在 WSBPEL 流程执行引擎上自动执行；在此基础上研发 WSBPEL 流程测试用例自动生成的支持工具，对 WSBPEL 流程进行自动化测试。

对于 WSBPEL 流程测试用例的自动生成，作者研究团队提出了基于 SXM 的 WSBPEL 流程测试用例生成技术[32]，解决了以下关键问题。

1)测试模型的选取

研究团队提出用 SXM 作为 WSBPEL 流程的测试模型，益处体现在下述 3 个方面：

(1)SXM 可以建模 WSBPEL 流程中活动的执行路径和所有 Web 服务激活的顺序(通过将活动调用的伙伴 Web 服务的操作建模为处理函数和状态之间的迁移予以体现)；

(2)SXM 可以建模 WSBPEL 流程中传递和流转的数据(通过内存值的改变展现 BPEL 流程变量值的变化)；

(3)基于 SXM 可以自动产生测试用例集、测试数据，输出结果易于验证，因此测试工作可以自动执行。

2)测试模型的构建

提出由 WSBPEL 流程自动构造测试模型 SXM 的算法，使得测试模型的构造可以实现自动化。为满足 SXM 测试条件，在构造 SXM 的过程中，文献[32]给出了具体的解决方案。

3)测试用例的生成

在满足一定测试条件的情况下，文献[32]提出了基于 Stream X-machine 的测试用例生成方法，所以，基于 SXM 可以自动生成 WSBPEL 流程活动执行的测试路径。例如对于 loan approval 流程，基于 SXM 生成的测试路径如图 3 所示，其中，测试路径中处理函数的定义见文献[36]。

$p_1=\phi_1\phi_2$ $p_2=\phi_1\phi_2'$ $p_3=\phi_1\phi_3'$ $p_4=\phi_1\phi_2\phi_3$ $p_5=\phi_1\phi_2\phi_3\phi_4$ $p_6=\phi_1\phi_2\phi_3\phi_6$
$p_7=\phi_1\phi_2\phi_3\phi_4'$ $p_8=\phi_1\phi_2\phi_3\phi_6'$ $p_9=\phi_1\phi_2'\phi_4$ $p_{10}=\phi_1\phi_2'\phi_6$
$p_{11}=\phi_1\phi_2'\phi_4\phi_5$ $p_{12}=\phi_1\phi_2'\phi_4\phi_6$ $p_{13}=\phi_1\phi_2'\phi_6'\phi_5$ $p_{14}=\phi_1\phi_2\phi_3\phi_4\phi_5$
$p_{15}=\phi_1\phi_2\phi_3\phi_4'\phi_6$ $p_{16}=\phi_1\phi_2\phi_3\phi_6\phi_7$ $p_{17}=\phi_1\phi_2\phi_3\phi_6'\phi_4$ $p_{18}=\phi_1\phi_2'\phi_4\phi_5\phi_6'$
$p_{19}=\phi_1\phi_2'\phi_6'\phi_4\phi_5$ $p_{20}=\phi_1\phi_2\phi_3\phi_4\phi_5\phi_6$ $p_{21}=\phi_1\phi_2\phi_3'\phi_4'\phi_6\phi_7$
$p_{22}=\phi_1\phi_2\phi_3\phi_6\phi_7'\phi_4$ $p_{23}=\phi_1\phi_2\phi_3'\phi_6'\phi_4\phi_5$ $p_{24}=\phi_1\phi_2\phi_3\phi_4\phi_5'\phi_6'\phi_8$
$p_{25}=\phi_1\phi_2\phi_3'\phi_4'\phi_6\phi_7\phi_8$ $p_{26}=\phi_1\phi_2\phi_3\phi_6\phi_7'\phi_4\phi_8$
$p_{27}=\phi_1\phi_2\phi_3'\phi_6'\phi_4\phi_5\phi_8$ $p_{28}=\phi_1\phi_2'\phi_4\phi_5'\phi_6'\phi_8$

图 3 loan approval 流程的测试路径

对于上述测试路径，为了测试路径的自动执行，需进一步研究为测试路径提供测试数据的技术。本文提出了测试路径中测试数据自动生成的解决方案。对于 WSBPEL 流程，其在引擎上的执行过程是半自动化的，即流程运行时，仅需人工为部分 Receive 活动和 Pick 活动提供测试数据；流程中的其余后继活动自动执行，没必要为它们提供测试输入数据，它们需要的测试输入数据会通过 WSBPEL 流程变量(赋值活动为 WSBPEL 流程变量赋值)提供。所以，对于测试路径的执行，仅需对测试路径中的部分 receive 活动和 pick 活动提供测试数据，本文分两种情况对此进行讨论。

1)对于伙伴进行 receive 或 pick 调用的实例化业务流程的活动，从相应 Web 服务单元测试的黑盒测试数据中选取测试数据触发相应活动。

2)对于 BPEL 流程中出现的 receive 或 pick 的调用，看其前驱是否是 assign 活动，若是，则该该活动的测试数据不用提供，由 assign 活动为其调用操作的输入消息赋值。

对于第 2)种情况，为了判断 receive 或 pick 活动的前驱是否是 assign 活动，可以查找 BPEL 程序中 assign 活动的后继活动，并将所有 assign 活动的后继活动存入符号表中，若 receive 或 pick 活动出现在表中，则说明该活动的前驱是 assign 活动。

查找 assign 活动的后继活动的算法步骤如下。

1)如果 assign 活动标签中没有定义 Source 属性，可以采用下面的方法查找其后继活动。

(1)如果存在这个活动的直接后继活动，那么直接后继活动就是该活动的后继活动。

(2)如果不存在直接后继活动，就查找该活动上层活动的下一个活动，直到找到某个活动或者直到上层活动为空。

2)如果标签中定义了 Source 属性，说明这个活动是并发活动的源活动，该活动的后继活动为定义了 Target 属性的活动。

这样，测试路径中的测试数据输入序列就可以自动生成了。例如，对于 loan process 流程的测试路径 P28，仅需为测试路径中的处理函数 ϕ_1 提供测试数据即可，提供的 XML 格式的测试数据文件如下所示。

```
<? xml version="1.0" encoding="UTF-8"?>
<loan:loanProcessRequest
  xmlns: loan = " http://schemas. active-endpoints. com/sample/
  LoanRequest/2008/02/loanRequest. xsd">
    <loan:loanType>Auto</loan:loanType>
    <loan:firstName>James</loan:firstName>
    <loan:lastName>Jones</loan:lastName>
    <loan:dayPhone>561</loan:dayPhone>
    <loan:nightPhone>999</loan:nightPhone>
    <loan:socialSecurityNumber>054871884</loan:socialSecurityNumber>
    <loan:amountRequested>10001</loan:amountRequested>
    <loan:loanDescription></loan:loanDescription>
    <loan:responseEmail></loan:responseEmail>
</loan:loanProcessRequest>
```

由于各测试路径中处理函数的内存变化，及相应的输出都体现在测试模型 SXM 中，因此，可以从处理函数的内存变化和输出，对 BPEL 流程活动执行结果的正确与否进行判断。

综上，对于图 2 所示的 Web 服务测试的技术框架，作者的研究团队提出了由规格说明 WSDL 自动构建操作输入消息测试类型模型树，为 Web 服务的单个操作自动生成了测试用例；提出了由语义 Web 服务 WSDL-S 自动构建状态 Web 服务的测试模型 Stream X-machine，为状态 Web 服务自动生成了操作序列的测试用例；将单独 Web 服务测试生成的测试用例，按照 2.1 节现有工具要求的测试用例描述的格式撰写，以自动化执行测试用例和管理 Web 服务测试过程；提出了由 WSBPEL 流程源码自动构建测试模型 SXM，为 WSBPEL 流程自动生成了测试路径和测试路径执行的测试数据，然后通过 WSBPEL 流程执行引擎自动执行相应的测试路径。

4 测试用例自动生成原型系统的实验

本文研制了 Web 服务测试用例自动生成原型系统包含的 4 个支持工具，即 OperTestCaseGen，SWSDSXMGen，BPELSXMGen和 DSXMTestPathGen，各工具的输入、输出、功能以及它们之间的关系如图 4 所示。

1)操作测试用例生成工具 OperTestCaseGen，输入为 WSDL 文件、相应叶子节点的测试用例集及用户选择的黑盒测试策略，输出为操作的测试用例集。

2)Web 服务测试模型 DSXM 构造工具 SWSDSXMGen，输入为操作对持久数据的改变信息和根据语义获取的操作模式值的属性信息，输出为 Web 服务的测试模型 DSXM。

3) WSBPEL 流程的测试模型 DSXM 构造工具 BPELSXMGen，输入为 WSBPEL 流程源码，输出为 WSBPEL 流程的测试模型 SXM。

4) 基于 DSXM 的合法测试路径集自动生成工具 DSXMTestPathGen，输入为 DSXM 测试模型，输出为合法的测试路径集。

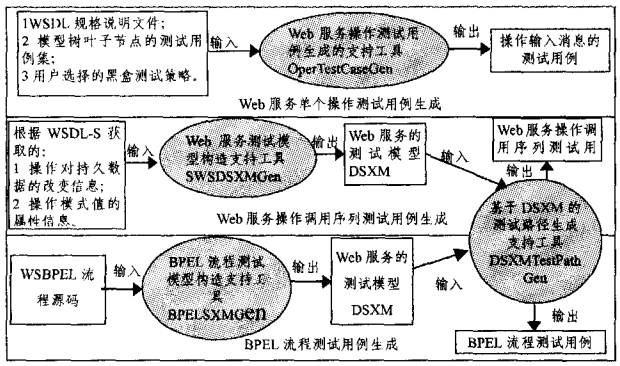

图 4 Web 服务测试用例自动生成原型系统

运用工具 OperTestCaseGen 可以为 Web 服务单个操作生成测试用例；运用工具 SWSDSXMGen 和 DSXMTestPathGen 为状态 Web 服务的操作序列自动生成测试用例；运用工具 BPELSXMGen 和 DSXMTestPathGen 为 Web 服务的组合流程 WSBPEL 自动生成测试用例。

表 1 给出了运用工具 OperTestCaseGen 为 Web 服务操作输入消息构造模型树的统计结果。表中信息包括 4 个 Web 服务案例对应操作输入消息模型树的节点数、构造模型树的耗时以及计算根节点与各叶子间关系的耗时。实验结果表明，输入消息模式树构造的耗时和计算根节点与各叶子间关系的耗时与构造的模型树节点数成正比。

表 1 OperTestCaseGen 解析 WSDL 案例的统计结果

	案例 1：Web 服务 Order 操作 order 的输入消息	案例 2：Web 服务 ThirdPartyCall 操作 makeCall 的输入消息[33]	案例 3：Web 服务 oogleSearch 操作 doGoogleSearch 的输入消息[34]	案例 4：Web 服务 Communication. wsdl 的操作 SendFileEx3 的输入消息[35]
模型树的节点数	16	16	12	14
构造模型树的耗时(ms)	15	14	10	11
计算根节点与叶节点间关系的耗时	6	5	2	4

表 2 给出了运用 BPELSXMGen 工具构造各个 BPEL 流程案例的统计结果。表中信息包括解析各 BPEL 流程案例生成 SXM 的状态数和耗时统计。实验结果表明，该构造工具的耗时与 BPEL 流程规模大小和生成 SXM 的状态数密切相关。

表 2 BPELSXMGen 解析流程案例的统计结果

	案例 1 loan approval 流程[30]	案例 2 shippingService 流程[30]	案例 3 auctionService 流程[30]	案例 4 SOAOrderBooking 流程(注：Oracle 公司提供的公开案例)
生成的 SXM 的状态数	12	10	14	48
耗时(ms)	47	46	64	187

表 3 给出了用工具 DSXMTestPathGen 为 4 个案例生成测试路径的相关信息，包括各案例对应的 DSXM 状态数、生成的路径数、最长路经的长度以及路径计算耗时。实验表明，计算耗时与各 DSXM 状态数以及 $V(q)$ 计算相关。经过大量反复的实验，发现计算 $V(q)$ 的耗时占主导因素。

表 3 DSXMTestPathGen 计算的案例程序的测试路径集的信息统计

	状态数	生成路径总数	最长路径的长度	耗时(ms)
案例 1：Web 服务 Mutimedia Conference 对应的 SXM	5	169	5	141
案例 2 loan approval 流程对应的 SXM	12	28	7	344
案例 3 shippingService 流程对应的 SXM	10	11	11	140
案例 4 auctionService 流程对应的 SXM	14	24	9	60485

结束语 在分析 Web 服务测试的层次和阶段的基础上，提出了基于模型的 Web 服务自动化测试的技术框架，并总结了作者团队在此技术框架内解决的关键技术；根据前期的研究成果，研制了测试用例自动生成原型，阐释了基于原型系统的实验结果。在目前研究工作的基础上，Web 服务自动化测试在以下 5 个方面值得进一步探讨。

1)为 XML Schema 内建数据类型测试用例的生成建立知识库或函数库

对于 XML Schema 而言，每种内建数据类型都定义了若干种类型的约束。对于每种内建数据类型，考虑建立在其上的各种约束。为各种约束测试用例生成的知识库或函数库建立内建数据类型，不但有利于 Web 服务操作测试用例的自动生成，也有利于应用于 XML Schema 数据类型的其它领域的软件测试。

2)为状态 Web 服务的测试路径自动生成测试数据

在为 Web 服务案例多媒体国际会议生产测试用例时，共生成 169 个测试路径，且各测试路径是自动生成的。在做故障注入实验时，对于各测试路径的执行，手工给予了测试路径中的各测试数据。测试数据的手工生成较繁锁、耗时，所以在文献[27]的基础上，可以对测试路径中的测试数据的自动生成进一步深入研究，可以从下面 3 个角度出发对该问题进行思考：

(1)测试路径就是 Web 服务的操作序列，各操作的测试数据可以用文献[20]中操作测试用例生成的方法产生，这样

就可以自动生成测试路径中的测试数据，但是还应该考虑(2)和(3)两种情况；

(2)由(1)获取测试路径中测试数据的方法没有考虑各操作间输入和输出的依赖关系，例如在某个测试路径中，上一个操作 o_i 的输出可能是下一个操作 o_j 的输入(其中 i 和 j 表示操作在测试路径中出现的索引，$j>i$)，那么操作 o_j 的测试数据就可以从 o_i 的输出获取，而不用采用(1)中的方法获取；

(3)当采用(1)中的方法为测试路径中某操作获取测试数据时，要考虑到保证该测试路径中后续操作都存在合法的输入数据，以保证该测试路径的执行。

3)研究测试模型 SXM 如何体现 BPEL 流程的异常处理机制

在现有研究的基础上，需要进一步研究如何用 SXM 的模型元素体现 BPEL 流程的故障和补偿处理机制，以支持变量之间的错误繁殖，测试整个流程故障活动的效果。

4)研究 Web 服务测试预言的自动生成

测试预言是一种检验待测系统在特定执行环境下是否正确运行的方法。期望结果是一种比较理想的测试预言，它用来确定测试用例执行的成功与否，是测试对象根据输入应该得到的输出。自动生成期望结果不仅能有效地减轻测试人员的负担，而且能为不间断的持续测试提供有力支持。但是现有的对自动生成期望结果的研究工作很少[76]。由于状态 Web 服务及其组合的测试模型 DSXM 体现了控制状态和内存值的变化，通过观察处理函数的输出和内存值的变换可以判断操作结果的正确性，因此，研究状态 Web 服务及其组合期望结果的自动生成是一个可行的方向。根据文献[27]生成的测试模型 DSXM、状态 Web 服务及其组合期望结果的自动生成也是我们正在进一步研究的课题。

5)研究 Web 服务回归测试方法

Web 服务的一个重要特点就是其动态升级性。Web 服务的使用者无法控制服务本身发生的改变，如果没能意识到服务发生的变化，就有可能导致发生不期望的系统故障，这对一些涉及安全或者可靠性要求高的系统尤其重要。这就需要一个系统集成者对其使用的 Web 服务定期进行重新测试，检查其是否满足：(1)功能需求；(2)非功能性质量需求。

Web 服务的回归测试是 Web 服务升级后，重新测试它的过程，以确保它的改变不会对其提供的功能有负面影响。问题的关键是 Web 服务提供者不知道哪些用户在以什么样的使用模式复用 Web 服务，因此当 Web 服务的接口或实现发生改变时，回归测试的责任就转移到 Web 服务的用户身上。如何基于 Web 服务升级的信息对其测试模型进行修改，以体现其升级的特性，以及如何重点对测试模型更新的部分进行回归测试，都是很好的研究方向。另外，针对 BPEL 流程的升级特性，具体基于 DSXM 的回归测试算法也有待于进一步研究。

参 考 文 献

[1] http://www.webinject.org/

[2] https://wstest.dev.java.net/

[3] http://www.soapui.org

[4] http://www.pushtotest.com

[5] http://www.webservices.org/highlights/optimyz_launches_webservicetester_3_0_a_web_services_testing_and_diganostics_solution/(go)/Articles

[6] http://www.adventnet.com.cn/products/qengine/web-services-testing.html

[7] Brenner D, Atkinson C. et al. Strategies for the Run-Time Testing of Third Party Web Services[C]// IEEE International Conference on Service-Oriented Computing and Applications(SOCA '07). June 2007:114-121

[8] Sinha A, Paradkar A. Model Based Functional Conformance testing of web services operating on persistent data[C]// Proceedings of the 2006 workshop on Testing, analysis, and verification of Web services and applications. ACM, New York, NY, USA, 2006:17-22

[9] Keum C S, Kang S, et al. Generating test cases for web services using extended finite state machine[J]. Lecture Notes in Computer Science, Springer Berlin/Heidelberg, 2006:103-117

[10] Whittaker J A. Stochastic software testing. Annals of Software Engineering[J]. Springer Netherlands, August 1997:115-131

[11] Whittaker J A. Software testing: What it is, and why it is so difficult[J]. IEEE Software, 1999

[12] El-Far I K I. Automated Construction of Software Behavior Models[D]. Florida Institute of Technology Melbourne, Florida, USA, 1999

[13] Arilo C, Neto D. A survey on model-based testing approaches: a systematic review[C]// WEASELTech '07. Atlanta Georgia, USA Copyright, ACM, 2007:31-36

[14] 颜炯等. 基于模型的软件测试综述[J]. 计算机科学, 2004, 31(2)

[15] [IEEEStd 610.12 — 1990]IEEE Standard Glossary of Software Engineering Terminology[S]. IEEE Standards Software Engineering, Volumeone: Custormerand Terminology Standards. 1999

[16] Ipate F, Holcombe M. Testing data processing-oriented systems from stream X-machine models[J]. Theoretical Computer Science, 2008, 403:176-191

[17] Ipate F. Testing against a non-controllable stream X-machine using state counting[J]. Theoretical Computer Science, 2006, 353:291-316

[18] Bogdanov K. Testing from X-Machine Specifications [C]// Hierons R M, et al., eds. Formal Methods and Testing, LNCS 4949. Springer-Verlag Berlin Heidelberg, 2008:184-208

[19] Bogdanov K, et al. Testing Methods for X-machines, a Review [J]. Formal Aspects of Computing, 2006, 18:3-30

[20] Ma Chun-yan, Du Cheng-lie, Zhang Tao, et al. WSDL-Based Automated Test Data Generation for Web Service[C]// International Conference on Computer Science and Software Engineering(CSSE 2008). IEEE Computer Society Press, 2008:731-737

[21] Tsai W T, Paul R, Wang Y, et al. Extending WSDL to Facilitate Web Services Testing[C]// Proceedings of the 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02). 2002:171-172

[22] Bertolino A, Polini A. The Audition Framework for Testing Web Services Interoperability[C]// Proceedings of the 2005 31st EUROMICRO Conference on Software Engineering and Advanced Applications(EUROMICRO-SEAA'05). IEEE Computer Society, 2005:134-142

[23] Heckel R, Mariani L. Automatic conformance testing of Web services[C]// Cerioli M, ed. FASE LNCS 3442. Springer-Verlag

Berlin Heidelberg,2005:34-48

[24] Martin D,Burstein M,Hobbs J,et al. OWL-S:Semantic Markup for Web Services[EB/OL]. http://www.w3.org/submission/2004/SUBM-OWL-S-2041122/

[25] Akkiraju R, Farrell J, Miller J, et al. Web Service Semantics-WSDL-S[R]. W3C Member Submission 7,November 2005

[26] Ipate F,et al. Testing(Stream) X-machines. Applicable Algebra in Engineering, Communication and Computing[M]. Springer-Verlag,2003:217-237

[27] Ma Chun-yan,Wu Jun-sheng,Zhang Tao. Web Service Sequence Testing Based on Stream X-machine[C]// The 10th International Conference on Quality Software(QSIC 2010). 2010,7:232-239

[28] Ipate F,Holcombe M. Testing data processing-oriented systems from stream X-machine models[J]. Theoretical Computer Science,2008,403:176-191

[29] Ipate F. Testing against a non-controllable stream X-machine using state counting[J]. Theoretical Computer Science, 2006, 353:291-316

[30] Alves A,et al. Web Services Business Process Execution Language Version 2.0,OASIS Standard[S]. April 2007. http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[31] Kavantzas N, Burdett D, Rizinger G, et al. Web Service Choreography Description Language Version 1.0[Z]. http://www.w3.org/TR/ws-cdl-10/,2005-11

[32] Ma Chun-yan,Wu Jun-sheng,Zhang Tao,et al. Testing BPEL with Stream X-Machine[C]// 2008 Internatinal Symposium on Information Science and Engineering(ISISE). 2008:578-582

[33] Open Service Access(OSA). Parlay X Web Services:Part 12:Multimedia Conference [OL]. http://www.parlay.org/en/specifications/pxws.asp

[34] http://api.google.com/GoogleSearch.wsdl

[35] http://seeker.dice.com/jobsearch/results/US/Detroit-Metro/Communication/w3r+Consulting/WSDL

[36] Ma Chun-yan,Wu Jun-sheng,Zhang Tao,et al. Automatic Test Case Generation for BPEL Using Stream X-machine[J]. International Journal of u- and e-Service, Science and Technology, 2008,1(1):27-35

(上接第 161 页)

A[] not deadlock:通过验证表明系统能够正常运行,不存在死锁;

E<>GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_Getdataover:通过验证表明系统能够正常传输数据;

E<>GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_Timeoutstate:通过验证表明数据传输中可能存在超时;

A<>GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_Timeoutstate imply GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_idle:

通过验证表明在传输数据过程中如果出现超时,系统总是能够复位;

E<>GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_pre_Errtranstate:通过验证表明数据传输中可能存在出现错误的情况;

A<>GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_pre_Errtranstate imply GNC_impl_GetGyrodatasys.GNC_impl_GetGyrodatasys_idle:

通过验证表明在传输数据过程中如果出现数据错误,系统总是能够复位。

图 6 UPPAAL 下 GNCC 对陀螺取数行为的验证

2.2.3 局限性分析

我们通过建立 AADL 行为模型与 UPPAAL 下时间自动机模板之间的映射关系,实现两种模型之间的自动转换;采用形式化的方法验证构件行为,增加了验证 AADL 模型的途径。但是由于两种模型之间存在一些差异,模型转换中的一些语义、性质可能丢失,如实型变量无法在 UPPAAL 下表示等,因此现阶段在构建 AADL 行为模型时,需要避免使用实型变量等情况,从而实现模型转换中语义、性质的保持。原型工具 AADLBA2UPPAAL 在对〈connections〉块、子程序调用及层次状态机的转换方面还有待实现,且只能够对单一模型文件进行转换,需进一步完善。

结束语 本文通过分析 AADL 行为附件文法的语义,建立了 AADL 行为模型与 UPPAAL 时间自动机模板之间的转换映射关系,实现了模型转换的原型工具。根据某型号航天器的需求,构建了 GNCC 对陀螺取数的 AADL 行为模型,通过工具自动转换为时间自动机模型,并在 UPPAAL 下对其取数过程的协议设计进行了验证。通过验证表明,数据在 GNCC 与陀螺之间能够正常地发送和接收;在遇到数据丢失及校验错误的情况下也能够恢复到初始状态,继续进行下一次数据传输。

AADL 行为附件的提出增强了 AADL 语言对构件内部行为的描述能力。本文利用形式化的方法对 AADL 行为模型进行行为验证,可信度较高。但是模型转换过程中语义、性质的一致性保持及证明还有待进一步研究。

参 考 文 献

[1] SAE Aerospace. SAE AS5506A:Architecture Analysis and Design Language V2.0[EB/OL]. http://www.sae.org/technical/standards/AS5506A,2009

[2] 刘倩,桂盛霖,李允,等. 基于 UPPAAL 的 AADL 模型可调度性验证[J]. 计算机应用,2009,29(7):1820-1824

[3] SAE Aerospace. SAE AS5506 Annex: Behavior_Specification V1.6[EB/OL]. http://www.aadl.info/aadl/documents/Behaviour_Annex1.6.pdf,2006

[4] Yang Z B,Hu K,Ma D F,et al. Towards a formal semantics for the AADL behavior annex[C]//Proceedings of the IEEE/ACM Design, Automation and Test in Europe 2009. Nice: EDAA, 2009:1166-1171

[5] França R B,Bodeveix J P,Filali M,et al. The AADL behaviour annex-experimentsand roadmap[C]// Proceedings of the 12th IEEE Int'l Conf. on Engineering Complex Computer Systems. Washington:IEEE Computer Society,2007:377-382

[6] Chkouri M Y,Robert A,Bozga M,et al. Translating AADL into BIP-Application to the verification of real-time systems[C]// Proceedings of the ACESMB 2008 Workshop Conjunction with MODELS 2008. Berlin,Heidelberg:Springer-Verlag,2009:5-19

[7] 杨志斌,皮磊,胡凯,等. 复杂嵌入式实时系统体系结构设计与分析语言:AADL[J]. 软件学报,2010,21(5):899-915

[8] Behrmann G,David A,Larsen K G. A tutorial on UPPAAL[C]// Proceedings of the 4th Int'l School on Formal Methods for the Design of Computer, Communication, and Software Systems. Bertinoro:Springer-Verlag,2004:200-236