Journal of Computer Applications
计算机应用,2012,32(8):2338 - 2341
ISSN 1001-9081
CODEN JYIIDU
2012-08-01
http://www.joca.cn

文章编号:1001 - 9081(2012)08 - 2338 - 04
doi:10.3724/SP.J.1087.2012.02338

基于程序依赖图的静态 BPEL 程序切片技术

王洪达[1*],邢建春[1],宋 巍[2],杨启亮[1]
(1.解放军理工大学 工程兵工程学院,南京 210007; 2.南京理工大学 计算机科学与技术学院,南京 210094)
(* 通信作者电子邮箱 wanghongda000@126.com)

摘 要: 传统程序切片技术在计算 BPEL 程序切片时会产生切片不完备问题,为此,提出一种基于程序依赖图的 BPEL 静态程序切片技术。该技术根据 BPEL 语言的特点,通过建立 BPEL 程序依赖图,计算 BPEL 程序切片。案例分析表明,该技术能够获得更加全面的程序切片,从而可以帮助软件工程人员更好地测试、调试和维护 BPEL 程序。

关键词: Web 服务组合; BPEL 程序依赖图; 静态程序切片; 异步调用依赖
中图分类号: TP311.5 **文献标志码:** A

Static BPEL program slicing technique based on BPEL program dependence graphs

WANG Hong-da[1*], XING Jian-chun[1], SONG Wei[2], YANG Qi-liang[1]
(1. *Engineering Institute of Engineer Corps, PLA University of Science and Technology, Nanjing Jiangsu* 210007, *China*;
2. *School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing Jiangsu* 210094, *China*)

Abstract: The slices of BPEL obtained are not complete if traditional program slicing technique is used. Therefore, a static BPEL program slicing technique based on BPEL program dependence graphs was proposed. This technique computed slices based on BPEL program dependence graphs, which were created according to the characteristics of BPEL. The results of case analysis prove that the technique based on BPEL program dependence graphs can obtain more complete slices, and thus giving software engineers more help to test, debug and maintain BPEL programs.

Key words: Web service composition; BPEL program dependence graph; static program slicing; asyn-invocation dependence

0 引言

Web 服务作为一种新兴的 Web 应用模式,是一种崭新的分布式计算模型,其目的是要解决异构平台上数据和应用的整合与共享问题。单个 Web 服务可能只提供唯一的调用函数来完成一个单一的功能,而将多个 Web 服务进行有机组合从而能完成一系列复杂任务。面向服务架构(Service Oriented Architecture,SOA)是当前 Web 服务的一种最主要的使能模型,其本质特征在于"松耦合",服务的发布、查询和使用呈现三阶段松耦合结构,以适合开放、动态、难控环境的需要[1-2]。SOA 以服务组合为重要特性。目前,BPEL4WS[3](简称 BPEL)已成为 Web 服务组合编程语言的业界事实标准。因此,保证 BPEL 程序的正确性和质量成为一个研究热点。

然而,Web 服务的松耦合性和分布性为 Web 服务组合语言 BPEL 的编程、理解和测试带来了巨大的挑战[4-6]。程序切片是一种用于分解程序的程序分析技术,广泛用于辅助软件工程人员测试、调试和维护软件[7-8]。程序切片技术经历了从静态到动态、从前向到后向、从单一过程到多个过程、从非分布程序到分布式程序、从基于系统依赖图的程序切片到面向对象的程序切片等几个过程。目前切片技术在软件分析、程序调试、集成、维护、测试、度量、软件重用和逆向工程等方面都有着广泛的应用。随着面向对象程序设计语言和设计方法的兴起,国内外已经提出了一些面向对象的程序切片方法,例如文献[9]通过利用类依赖图来计算C++程序的切片,文献[10]利用对象依赖图来切片面向对象程序等。但是,传统的程序切片技术在用于 BPEL 程序切片时会产生切片不完备等问题。因此,本文根据 BPEL 语言的特点(例如并发、同步等),提出一种基于程序依赖图的 BPEL 静态程序切片技术,从而可以帮助软件工程人员更好地理解和测试 Web 服务组合中的 BPEL 程序。案例分析表明,与现有切片技术相比,本文提出的切片技术能够获得更加全面的程序切片,从而可以帮助软件工程人员更好地测试、调试和维护 BPEL 程序。

1 相关概念

由于本文介绍的 BPEL 程序切片技术是基于 BPEL 程序依赖图的,因此首先介绍一些与 BPEL 程序依赖有关的基本概念,以便于读者更好地理解第 2 章将要提出的切片技术。

引入线性控制流图(Threaded Control Flow Graph,TCFG)[11]来帮助分析 BPEL 程序中的依赖关系。TCFG 拓展了传统的控制流图,主要是增加了 flow 和 end-flow 两个节点,用来表示 BPEL 流程中的 <flow>(并发)节点。

定义 1 后支配节点。在 TCFG 中,如果从节点 i 到结束(exit)节点的所有路径都必须经过节点 j,则称 j 是 i 的后支配

收稿日期:2012-02-27;修回日期:2012-03-22。
基金项目:国家自然科学基金资助项目(61003019,61073031);高等学校博士学科点专项科研基金资助项目(20113219120021)。
作者简介:王洪达(1987 -),男,河北沧州人,硕士,CCF 会员,主要研究方向:服务计算、软件测试; 邢建春(1964 -),男,河北石家庄人,教授,博士生导师,博士,CCF 高级会员,主要研究方向:计算机测量控制; 宋巍(1981 -),男,山东日照人,讲师,博士,CCF 会员,主要研究方向:软件工程与方法学、程序分析、服务计算; 杨启亮(1975 -),男,河南信阳人,讲师,硕士,CCF 会员,主要研究方向:计算机软件与理论、分布式控制系统。

节点。

定义 2　控制依赖。在 TCFG 中，节点 n_j 控制依赖于节点 n_i，如果满足:

1) n_i 和 n_j 之间存在一条路径 p(不包括节点 n_i 和 n_j) ，且 n_j 是 p 中除 n_i 以外所有节点的后支配节点;

2) n_j 不是 n_i 的后支配节点。

定义 3　数据依赖。在 TCFG 中，称节点 n_j 数据依赖于节点 n_i，如果满足:

1) 节点 n_i 的输出量是节点 n_j 的输入量;

2) n_i 和 n_j 之间存在一条路径 p，且在路径 p 中除了 n_i 节点，其他任何节点(不包括节点 n_i 和 n_j) 的输入量都不是 n_j 的输入量。

控制依赖和数据依赖的分析方法可以在文献 [12] 中获得。除了控制依赖和数据依赖，笔者曾在文献 [13] 中提出了一种新的依赖关系。这种依赖关系是由于 BPEL 流程中的异步调用机制所产生的。例如，在 BPEL 流程中，一个单向调用 < invoke > 活动后面必然有一个 < receive > 活动接收单向调用的结果。如果把两个活动调换顺序之后，将会引起死锁，所以这两个活动之间存在一种依赖关系。但是，这两个活动之间的依赖关系既不是控制依赖又不是数据依赖，本文称之为异步调用依赖。

定义 4　异步调用依赖[13]。在 TCFG 中，节点 n_j 异步调用依赖于 n_i，如果满足: n_j 代表 < receive > 活动，n_i 代表单向调用 < invoke > 活动，n_j 接收 n_i 的调用结果。

定义 5　BPEL 程序依赖图(BPEL Program Dependence Graph，BPDG) 。BPEL 程序依赖图是一个有向图 $\langle N, E\rangle$，其中:

1) N 为节点。其中有一个特殊的节点，即开始节点 entry，其余的节点代表语句和谓词表达式。

2) $E \in N \times N$，为有向边，$\langle n_i, n_j\rangle \in E$ 为一条从 n_i 到 n_j 的有向边，表示两个活动或者谓语表达式之间的控制依赖、数据依赖或者异步调用依赖关系。

对于程序依赖图，通常情况下只定义一个 entry 节点作为程序的入口节点，它与程序中语句的关系为控制依赖。

2　静态切片技术

计算静态程序切片的方法主要有两种: 根据数据流方程计算和根据依赖图关系计算。本文是通过根据已有的 BPEL 流程创建 BPEL 程序依赖图，并基于 BPEL 程序依赖图计算程序切片。

2.1　BPEL 程序依赖图

构造 BPDG 的基本思想是: 首先根据 BPEL 程序得到 BPEL 控制流图(Control Flow Graph，CFG) ，然后分别分析 BPEL 程序中语句与语句之间或语句与谓词表达式之间的控制依赖关系、数据依赖关系和异步调用依赖关系，最后得到 BPDG。

从 BPEL 程序生成 BPEL 控制流图可以参考文献 [13]，由于不是本文的重点，在此不再赘述。由于本文的方法是基于程序依赖图的，因此，首先介绍一下如何获得 BPEL 程序依赖图，具体步骤如下所示。

1) 控制依赖分析。一个 BPDG 中，节点 n_i 到节点 n_j 之间存在控制依赖关系当且仅当满足下列两个条件之一时:

①n_i 为谓词表达式且 n_j 为谓词表达式 n_i 的从属语句。如果 n_i 为循环谓词表达式(< repeatUntil > 或者 < while >) ，则边 $\langle n_i, n_j\rangle$ 标记为 true; 如果 n_i 为选择谓词表达式(< if > 或者 < pick >) ，则边 $\langle n_i, n_j\rangle$ 标记为 true 或者 false 分别取决于 n_j 出现在 then(< condition > 或者 < onMessage >) 支路还是 else(< else > 或者 < onAlarm >) 支路。

②当 n_i 为 entry 节点且 n_j 不从属于任何控制谓词表达式时，边 $\langle n_i, n_j\rangle$ 标记为 true。

需要注意的是，由于 BPEL 程序中 < flow > 的存在，需要引入一条同步依赖边。它主要是连接两个并行程序各自第一个节点和最后一个节点，以表明这两个 BPEL 程序是同步关系。

2) 数据依赖分析。数据依赖可以分为以下三种类型: 定义—使用(def-use) 依赖、使用—定义(use-def) 依赖和定义—定义(def-def) 依赖。由于后两种依赖可以通过更换变量名称来消除，所以在此只关注 def-use 依赖。可以根据变量名或者变量名的数据结构来确定数据依赖关系。

3) 异步调用依赖分析。可以将 BPEL 中的标签 < partnerLink > 和 < correlation > 与 TCFG 中的节点对应起来。如果两个活动 < invoke > 和 < receive > 的 < partnerLink > 和 < correlation > 相同，且 < receive > 活动作为 < invoke > 活动的后继节点，则这两个活动之间存在异步调用依赖关系。通过遍历 TCFG，可以得到所有的异步调用依赖关系。

4) 建立 BPEL 程序依赖图(BPDG) 。通过以上分析，可以将上述三种依赖关系结合 BPEL 控制流图，生成 BPEL 控制依赖图 BPDG。

2.2　BPEL 静态程序切片算法

定义 6　一个程序 P 的静态切片是由 P 中的一些语句和谓词组成的集合，这些语句和谓词影响在兴趣点 n 定义或使用的变量 v 的值(即静态向后切片) ，或者是那些受 n 点变量 v 的值影响的所有语句和谓词(即静态向前切片) 。

静态切片技术与程序的输入无关，它是一个可执行程序，按照一定的准则从源程序中移去零条或多条语句来构造。其准则表示方法为: $C = \langle q, v\rangle$，q 是程序中的一条语句，即兴趣节点，v 是一个程序变量。

静态程序切片包含前向切片(forward slicing) 、后向切片(backward slicing) 和双向切片(bidirectional slicing) 。例如，当一个程序员发现 BPEL 程序中某个变量的值错误，想要测试这个变量时，可以使用前向切片。这个切片将计算这个 BPEL 程序中在此变量之前的所有与这个变量有关的语句，从而可以帮助程序员更快地进行 BPEL 程序的测试; 同样，当一个程序员想要删除 BPEL 程序中某个变量时，可以使用后向切片，这样可以确保程序员能删除所有与此变量有直接关系或者间接关系的语句。

本文主要关注如何在 Web 服务组合的 BPEL 程序中产生静态程序切片。由于前向切片和后向切片的计算方法相似，所以主要介绍后向程序切片算法，前向程序切片算法和双向程序切片算法可以根据后向程序切片算法类比得到。基于前面介绍的 BPDG，静态后向程序切片可以经由一个切片标准开始，通过后向遍历 BPDG 得到。基于 BPDG 的静态后向切片算法如下所示。

输入: Web 服务组合 BPEL 程序的 BPDG; 切片标准 $C = (num, var)$;

输出: 静态后向切片集合 S。

```
Begin
  marked_list = {num};
  work_list = {num};
  while work_list ≠ ∅ do
```

```
    select a node u from work_list;
    work_list = work_list + {u};
    for each node v ∈ V(G) and (v, u) ∈ E(G) do
      if v ∉ marked_list then
        work_list = work_list ∪ {v};
        marked_list = marked_list ∪ {v};
      endif
    endfor
  endwhile
  S = marked_list;
  return S;
End
```

3 案例分析

通过一个广泛使用的服务组合的例子: BusinessTravelProcess[14] 来说明 BPDG 的建立以及如何基于 BPDG 使用静态切片技术。这个 Web 服务系统的主要功能是: 客户可以根据自己的需求选择合适的飞机票。这个例子主要包含三个服务: Employee Travel Status Web service、the American Web service 和 Delta Airlines Web service。这个 BPEL 流程的主体如下所示:

```
<process name = "BusinessTravelProcess"... >
  <variables > <variable name = "TrvlReq" .../ >
    <variable name = "EmplTrvlStReq" .../ >
    <variable name = "EmplTrvlStResp" .../ >
    <variable name = "FlightDetails" .../ >
    <variable name = "FlightRespAA" .../ >
    <variable name = "FlightRespDA" .../ >
    <variable name = "TrvlResp" .../ > </variables >
  <sequence >
    01  <receive partnerLink = "client" operation =
          "TrvlApproval" variable = "TrvlReq"... / >
    02  <assign > <copy > <from variable = "TrvlReq" part =
          "employee"/ > <to variable = "EmplTrvlStReq" part =
          "employee"/ > </copy > </assign >
    03  <invoke partnerLink = "emplTrvlSt" operation =
          "EmplTrvlStatus" inputVariable = "EmplTrvlStReq"
          outputVariable = "EmplTrvlStResp "... / >
    04  <assign > <copy > <from variable = "TrvlReq" part =
          "flightData"/ > <to variable = "FlightDetails" part =
          "flightData"/ > </copy > <copy > <from variable =
          "EmplTrvlStResp" part = "travelClass"/ >
          <to variable = "FlightDetails" part = "travelClass"/ >
          </copy > </assign >
    05  <flow >
    06  <sequence > <invoke partnerLink = "AAirl"
          operation = "FlightAvailability"
          inputVariable = "FlightDetails".../ >
    07  <receive partnerLink = "AAirl" operation =
          "FlightTicketCallback" variable = "FlightRespAA"
          .../ > </sequence >
    08  <sequence > <invoke partnerLink = "DAirl"
          operation = "FlightAvailability" inputVariable =
          "FlightDetails" / >
    09  <receive partnerLink = "DAirl" operation =
          "FlightTicketCallback" variable = "FlightRespDA" / >
          </sequence >
    10  </flow >
    11  <switch > <case condition =
          "bpws: getVariableData( ´FlightRespAA´,
          ´confData´,´/confData/Price´) =
          bpws: getVariableData( ´FlightRespDA´,
          ´confData´,´/confData/ Price´) " >
    12  <assign > <copy > <from variable =
          "FlightRespAA" / > <to variable = "TrvlResp" / >
          </copy > </assign > </case >
    13  <otherwise > <assign > <copy > <from variable =
          "FlightRespDA" / > <to variable = "TrvlResp" / >
          </copy > </assign > </otherwise >
    14  </switch >
    15  <invoke partnerLink = "client" operation =
          "ClientCallback" inputVariable = "TrvlResp".../ >
  </sequence >
</process >
```

根据第 2 章的分析,可以通过如下步骤实施静态技术切片技术。

1) 控制依赖分析。通过分析,可以建立如图 2 所示 BPEL 控制依赖图。为了简洁,省略了没有执行的节点,例如 10 和 14。

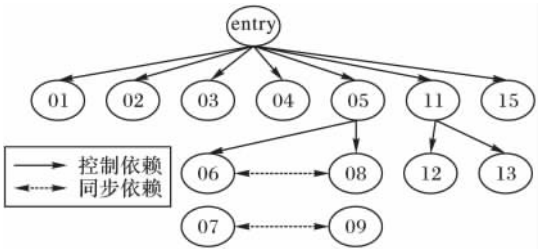

图 2 BPEL 控制依赖图

2) 数据依赖分析。根据上章所述,可以得到如表 1 所示的变量之间的 def-use 依赖关系。

表 1 BPEL 程序中每条语句的 def-use 变量

编号	定义(def)	使用(use)
01	TrvlReq	
02	EmplTrvlStReq. employee	TrvlReq. employee
03	EmplTrvlStResp	EmplTrvlStReq
04	FlightDetails. flightData FlightDetails. travelClass	TrvlReq. flightDat EmplTrvlStResp
06		FlightDetails
07	FlightRespAA	
08		FlightDetails
09	FlightRespDA	
11		FlightRespAA, FlightRespDA
12	TrvlResp	FlightRespAA
13	TrvlResp	FlightRespDA
15		TrvlResp

根据表 1 的数据依赖分析,图 2 所示 BPEL 程序中的所有变量的数据依赖关系如表 2 所示。

表 2 BPEL 程序中所有变量之间的数据依赖

变量	数据依赖
TrvlReq	(01, 02), (01, 04)
EmplTrvlStReq	(02, 03)
EmplTrvlStResp	(03, 04)
FlightDetails	(04, 06), (04, 08)
FlightRespAA	(07, 11), (07, 12)
FlightRespDA	(09, 11), (09, 13)
TrvlResp	(12, 15), (13, 15)

3) 异步调用依赖分析。结合异步调用依赖,通过分析 BusinessTravelProcess[14] 的 BPEL 程序,可以很直观地发现如

下异步调用依赖: 07 异步调用依赖于 06,09 异步调用依赖于 08。

4) 建立 BPEL 程序依赖图(BPDG) 。通过如上所述分析,可以建立如图 3 所示的 BPEL 程序依赖图。

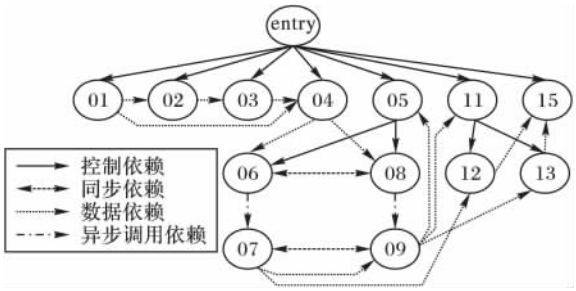

图 3　BPEL 程序依赖图

基于 BPDG,可以使用树的遍历法来求得 BPDG 的静态前向切片、静态后向切片和静态双向切片。例如: 当选定以(15,TrvlResp) 做前向切片时,利用图 3 得到的 BPDG,可以得到如图 4 所示的静态前向切片。

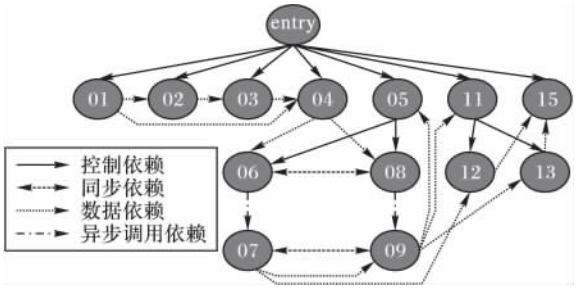

图 4　以(TrvlResp,15) 为标准的 BPEL 静态前向切片

如图 4 所示,灰色的形状(01,02,03,04,05,06,07,08,09,11,12,13) 即标准为(15,TrvlResp) 的静态前向切片。

而现有的 BPEL 静态切片技术[15-16] 由于没有考虑到异步调用依赖,其以标准为(15,TrvlResp) 的静态前向切片如图 5 所示。

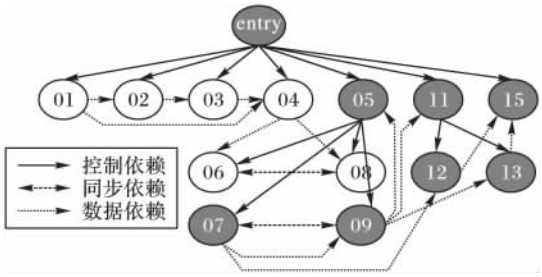

图 5　以(TrvlResp,15) 为标准的现有技术的 BPEL 静态前向切片

如图 5 所示,灰色的形状(07, 09,11,12,13,15) 即标准(TrvlReq,01) 为静态前向切片。

从这个案例中可以很明显地看出,本文提出的 BPEL 静态切片技术比现有静态切片技术更全面。

4　结语

Web 服务的分布性和松耦合性是其作为分布式计算的一大优势,然而,这种特性也给程序员理解、编写和测试 Web 服务组合语言 BPEL 带来了巨大的挑战。为此,根据 BPEL 的特点,本文提出了一种基于程序依赖图的 BPEL 静态程序切片技术。案例分析表明,该技术能够获得更加全面的切片技术,从而可以帮助软件工程人员更好地测试、维护 BPEL 程序。下一步打算对 Web 服务组合做进一步的动态切片和精确切片,并将其应用到 Web 服务的回归测试中。

参考文献:

[1] 吕建, 马晓星, 陶先平, 等. 网构软件的研究与进展[J]. 中国科学 E 辑, 2006, 36(10): 1037 - 1080.

[2] 宋巍,唐金辉,张功萱,等. WS-BPEL 服务可替换性分析[J]. 中国科学: 信息科学, 2012,42(3): 264 - 279.

[3] OASIS Web Services Business Process Execution Language. Web services business process execution language version 2.0 [EB/OL]. [2011 - 12 - 06]. http://docs.oasis-open.org/wsbpel/2.0/CS01/wsbpel-v2.0-CS01.pdf.

[4] LI L, CHOU W, GUO W P. Control flow analysis and coverage driven testing for Web services [C]// ICWS '08: Proceedings of the 2008 IEEE International Conference on Web Services. Piscataway: IEEE, 2008: 473 - 480.

[5] LI L, CHOU W, GUO W P. An abstract GFSM model for optimal and incremental conformance testing of Web services [C]// ICWS '09: Proceedings of the IEEE International Conference on Web Services. Piscataway: IEEE, 2009: 205 - 212.

[6] YUAN YUAN, LI ZHONGJIE, SUN WEI. A graph-search based approach to BPEL4WS test generation [C]// International Conference on Software Engineering Advances. Piscataway: IEEE, 2006: 205 - 212.

[7] RAY M, KUMAWAT K I, MOHAPATRA D P. Source code prioritization using forward slicing for exposing critical elements in a program [J]. Journal of Computer Science and Technology, 2011, 26(2): 314 - 327.

[8] 李必信. 程序切片技术及其应用[M]. 北京: 科学出版社, 2006.

[9] LARSEN L, HARROLD M J. Slicing object-oriented software [C]// ICSE '96: Proceedings of the 18th International Conference on Software Engineering. Washington, DC: IEEE Computer Society, 2006: 495 - 505.

[10] ZHAO JIANJUN, CHENG JINGDE, USHIJIM K. Static slicing of concurrent object-oriented programs [C]// COMPSAC '96: Proceedings of the 20th Conference on Computer Software and Applications. Washington, DC: IEEE Computer Society, 2006: 312 - 320.

[11] NANDA M G, CHANDRA S, SARKAR V. Decentralizing execution of composite Web services [C]// OOPSLA '04: Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. New York: ACM, 2004: 170 - 187.

[12] FERRANTE J, OTTENSTEIN K J, WARREN J D. The program dependence graph and its use in optimization [J]. ACM Transactions on Programming Languages and Systems, 1987, 9(3): 319 - 349.

[13] SONG WEI, MA XIAOXING, CHEUNG S C, *et al*. Refactoring and publishing WS-BPEL processes to obtain more partners [C]// IEEE International Conference on Web Services. Piscataway: IEEE, 2011: 129 - 136.

[14] MAO CHENGYING. Slicing Web service-based software [C]// IEEE International Conference on Service-Oriented Computing and Applications. Piscataway: IEEE, 2009: 1 - 8.

[15] YAN J, LI ZHONGJIE, YUAN YUAN, *et al*. BPEL4WS unit testing: Test case generation using a concurrent path analysis approach [C]// ISSRE '06: 17th International Symposium on Software Reliability Engineering. Piscataway: IEEE, 2006: 75 - 84.

[16] KEUM C, KANG S, KO I-Y, *et al*. Generating test cases for Web services using extended finite state machine [C]// TestCom'06: Proceedings of the 18th IFIP TC6/WG6.1 International Conference on Testing of Communicating Systems. Berlin: Springer-Verlag, 2006: 103 - 117.