

# 基于行为协议的构件软件静态测试研究

于素萍, 杨偕杰

(河南大学 计算机与信息工程学院, 河南 开封 475004)

**摘 要:**系统的静态分析能在设计开发阶段发现错误,从而避免了在运行时错误检测技术在系统执行期间带来的负面影响。基于尽可能避免静态错误这一构件测试策略的基本思想提出了一种对构件化软件系统进行静态测试的方法。采用通信模型对数据库服务构件系统进行抽象建模,并结合用于描述构件系统中构件之间交互的形式化方法行为协议,通过对构件系统构件行为协议的一致性验证,从而测试构件交互的正确性。

**关键词:**行为协议;静态测试;构件技术

**中图分类号:**TP311.5

**文献标识码:**A

**文章编号:**1673-629X(2008)03-0128-04

## Research of Component Static Testing Based on Behavior Protocols

YU Su-ping, YANG Xun-jie

(College of Computer and Information Engineering, Henan University, Kaifeng 475004, China)

**Abstract:** Static analysis of systems allows discovering errors at design time and to avoid run-time error detection techniques that negatively impact performance of the systems. The article is based on the extent possible, to avoid passive component testing the wrong strategy for the basic ideology of a pair of component-based software for the static test methods. In this paper, use the communication model of database services component systems to abstract modeling, and integrate components to describe the system of interaction between components formal methods acts agreement. It can formally validate the protocol conformance in component system.

**Key words:** behavior protocols; static testing; component technology

## 0 引言

如今,构件技术已经成为软件技术中重要的一部分。构件测试技术也日趋成熟,但是构件测试可能是构件技术中要求最为苛刻的一个方面。所以构件测试的方法,技术比较多,涉及的范围比较广,这些构件测试技术可以分为两大类,一是:构件化软件的静态测试;二是:构件化软件的动态测试。

文中是基于尽可能避免静态错误,这一构件测试策略<sup>[1]</sup>的基本思想上提出的一种对构件化软件系统进行静态测试的方法。构件是通过接口进行交互,在构件系统中只有构件正确的交互才能保证系统的正常运行。首先采用通信模型对数据库服务构件系统进行抽象建模,并结合文献<sup>[2]</sup>中提出的用于描述构件系统中构件之间交互的形式化方法行为协议,对构件的接口行为进行形式化的验证,从而讨论静态测试在构件化

软件系统的应用。

## 1 通信模型

文中抽象出一种特殊构件模型和 ADL<sup>[3]</sup>的依赖细节如命名空间、类型规则等。为了关注基本原则,模型是基于-agent概念的抽象构件,在这些构件中它们的接口变为 agent 之间的连接,接口上方法的调用转换为连接上的事件的发生;并且通过事件序列的跟踪模型一个构件的行为,而这些事件序列是建立在代表构件的 agent 的连接之上的;构件的行为能通过协议去分析和表达,这里定义的协议采用正则表示式的形式;在协议之上还定义了关系的概念,它允许静态的分析构件之间的交互。

在文中,agent 是一个能处理一系列事件的计算实体。由于要处理事件,agent 能发出事件,接受事件,并能处理内部事件,这些处理完全取决于 agent 的执行。agent 之间通信是通过端对端的双向连接转换事件,一个 agent 能和多个其他的 agent 进行通信,并且两个 agent 能通过多个连接进行。agent 可以是一个独立体,也可以是一个复合体,一个独立体 agent 它不包含任

收稿日期:2007-06-18

基金项目:河南省自然科学基金项目(0511011400);河南省教育厅自然科学基金项目(2004520014)

作者简介:于素萍(1981-),女,河南洛阳人,硕士研究生,主要研究方向为软件测试;杨偕杰,讲师,研究方向为软件设计、软件测试。

何的其它 agent,所有的连接都是外部的;一个复合的 agent 是由多个构件构造而成。假设 agent P 是一个复合 agent,它是有 agent A 和 agent B 组成的,A 和 B 的连接变成了 P 的连接,在这些连接之上的事件能被 P 处理,A 和 B 之间的连接变成了 P 的内部连接;A 和 B 其它连接变成了 P 的外部连接。文中把建立在 P 的内部连接之上的事件称为 P 的内部事件。P 的执行完全决定于它的内部连接和 A 和 B 的执行,图 1 显示了独立 agent A、B 和复合 agent P 和连接  $C_1, C_2, C_3, C_4$ 。A 和 B 是独立体,P 是由 A 和 B 构成, $C_1, C_2, C_3$  是 A 的外部连接,而  $C_2, C_4$  是 B 的外部连接; $C_1, C_3$  和  $C_4$  是 P 的外部连接, $C_2$  是 P 的内部连接,这个内部连接是 A 和 B 共享的连接。

在组成系统中,每个 agent 是系统中的一部分,根系统是一个没有外部连接的 agent。在系统中,所有的 agent 在收到初始化信号后开始处理事件;同样,所有的 agent 在收到停止信号后就停止事件的处理。

文中假设一个 agent 不能在同一时刻处理多个事件;没有连接延迟;一个 agent 发送一个事件仅当和它对应相连接的 agent 已经准备好去接收它。因此,在一个特定的运行系统中,建立一系列的连接 V 之上的活动的 agent 被观察为一个有限的事件序列,这些事件序列是有 A 在运行时处理的。按照惯例,这一系列的事件处理被看为是 A 在 V 上的轨迹,这个轨迹是一系列事件的标志,每一个标志代表了一个事件的精确处理。A 在 V 上所有可能的事件处理序列被认为是系统 M 中 A 在 V 上的行为。

为了要说明可重用的构件模型,希望能形式化地捕获 agent A 在任何系统和任何运行时的行为。为了达到这个目标,假设 A 的所有潜在相邻的 agents 也就是和 A 相连接的外部连接,形成了 A 的环境 E。换句话说,外部连接代表了 A 和 E 的接口。为了对具体的相邻 agent 进行抽象,简单假设在 A 和 E 之间有一个预定义的协议,这个协议是关于他们如何进行事件交流的。在一个设计合理的系统中,A 必须能吸收 E 发出的事件,同时 E 也必须能接受被 A 发出的事件。

为了能在更高抽象层次上规划合约,假设每个跟踪是提供方和需求方的一次交错,它代表一次 A 和 E 的特殊交流。这反映出了 A 提供服务给 E 同时,A 也能从 E 得到服务。因为形成 E 的 agent 不能清楚确定,假设为了区分在 A 的跟踪中的提供和需求,定义 A 的提供字符  $S_{prov}$ ,需求字符  $S_{req}$ ,作为一系列事件符号,用于代表 A 的外连接上的事件处理。为了方便,假定  $S_{ext} \cap S_{req} = \emptyset$ ,同时定义  $S_{prov} \cup S_{req}$  作为 A 的外部字符  $S_{ext}$ 。于是, $S_{ext}$  是 A 和 E 合约的一部分。

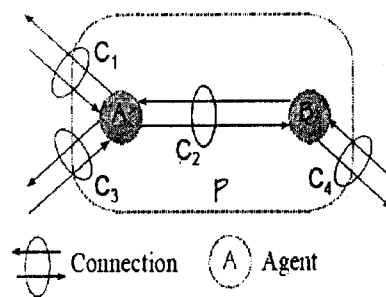


图 1 简单的构件模型

## 2 行为协议

对比 Wright<sup>[4]</sup>, TRACTA<sup>[5]</sup>, 选择了文献[2]提出的更容易理解、更易应用于构件体系结构规格描述和自动化工具的行为协议。

行为协议是建立在字符集 S 上的类似正则表达式,它产生基于 S 的跟踪。最简单的行为协议是一个事件符号或者是空标志,构建一个行为协议和构建一个正则表达式相似,并能使用操作符。基本的操作是那些正则表达式,增强型操作为描述并发,隐含的交流并很好地为无序响应提供了一个注释。行为协议的操作符定义如下:

基本的操作:

A;B 顺序:一系列的跟踪是由 A 产生的跟踪和 B 产生的跟踪连结而成的跟踪;

A+B 交替:一系列的跟踪是由 A 或者由 B 产生的;

A\* 重复:A 可以出现零次或者多次;

增强型操作:

A|B 与关系:A 和 B 产生的跟踪可以任意的交替;

A||B 或关系:表示  $A+B+(A|B)$ ;

A/G 限制:把不在集合 G 中的事件标示从 A 的跟踪中去除;

复合型操作:

$A \cap xB$  复合:如果 X 中包含的 x 在 A 中是以? x 在 B 中是以! x 出现,或者在 A 中是以! x 在 B 中是以? x 出现,则当出现? x,! x 或者是! x,? x 时要合并成  $\S x$ ;

$A | T | B$  调整:有一组轨迹,从每一对轨迹(a,b)形成任意交替的事件符,如果交替产生了...x,x...事件符,那么在最终的结果中 x,x 要合并成...x...。

## 3 建立行为模型

这一节将为一个数据库服务构件系统进行建模。这个简单的数据库服务,设计成为数据库构件 DB,一

个数据库模板的实例。DB 提供插入、删除和查询操作,进行数据库的插入记录、删除记录和查询记录。为了支持它的功能,DB 调用了另一个数据库构件实例为 DBAccess 构件的 Data 和实例为 LogMan 的构件 Login,它用于登陆操作。这些构件通过它们的提供接口发布它们的服务。DBAccess 通过 IDatabaseAccess 提供进入接口,LogMan 通过 ILogging 提供登陆验证接口。DB 通过 IDBServer 提供数据库服务接口。具体实例如图 2 所示。

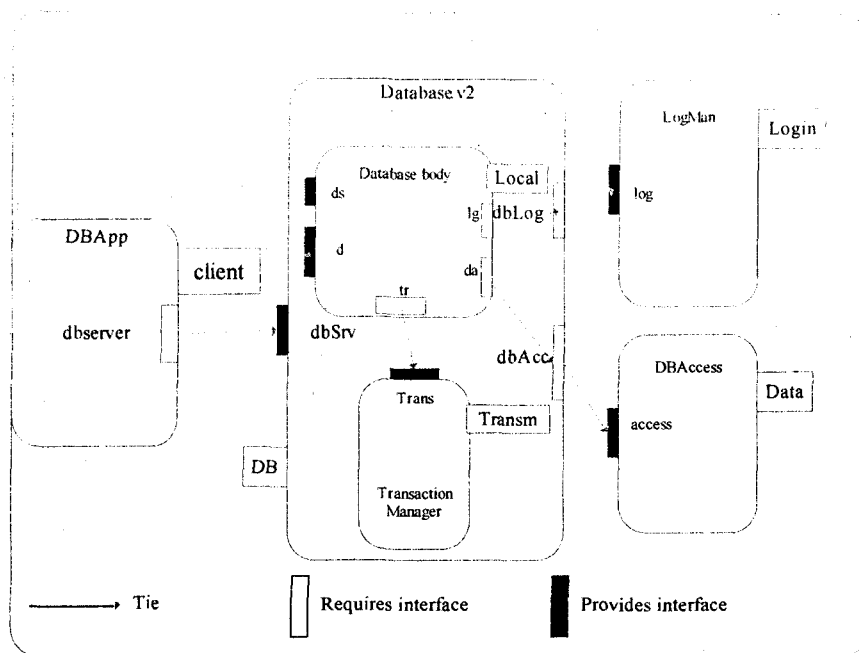


图 2 数据库服务通信模型

#### 4 形式化验证构件的行为

假设一个构件为一个模板  $T = \langle F, A \rangle$ ,  $F$  代表构件所在的框架,  $A$  代表构件的结构, 即此构件包含的子构件。在模板  $T$  中 dbAcc 的接口协议, 即 IDatabaseAccess 接口的实例必须遵守 DB 的框架协议。在验证协议一致性之前, dbAcc 在框架级别上的接口需求协议要来自于接口协议, 形式表示如下:

```
! dbAcc. Open ;
( ! dbAcc. Insert + ! dbAcc. Delete + ! dbAcc.
Query + ! dbAcc. GetTrModel + ! dbAcc. SetTrModel
) * ;
! dbAcc. Close
```

在本节中这个协议记为  $P_1$ , 它必须满足 DB 的框架协议。虽然这个顺应关系是基于语言限定的, 但是文中要用协议限定代替语言限定。因为它们是语义级别的, 并允许直接的协议比较。因此, 框架协议限定接口 dbAcc (记为  $P_F/I$ ) 采用下列形式:

```
! dbAcc. Open ;
( ! dbAcc. Insert * + ! dbAcc. Delete * + !
dbAcc. Query * ) * ;
! dbAcc. Close
```

$P_1$  必须在外部字符集  $S_{ext} = \{! dbAcc. Open?, ! dbAcc. Insert?, ! dbAcc. Delete?, ! dbAcc. Query?, ! dbAcc. GetTrModel?, ! dbAcc. SetTrModel?, ! dbAcc. Close?, ? dbAcc. Open?, ? dbAcc. Insert?, ? dbAcc. Delete?, ? dbAcc. Query?, ? dbAcc. GetTrModel?, ?$

dbAcc. SetTrModel?, ? dbAcc. Close?} 上遵循  $P_F/I$ , 按照协议一致性定义<sup>[2]</sup>, 仅仅需要验证包含  $L(P_F/I) \cdot L(P_1)$ 。考虑这个包含, 每一个来自于  $L(P_F/I)$  的轨迹  $t$  都以 dbAcc. Open 请求开始, 并跟随一系列的 dbAcc. Insert, dbAcc. Delete or dbAcc. Query 请求,  $t$  的最后两个事件符代表了请求 dbAcc. Close。当所有来自  $L(P_1)$  的轨迹跟随这个形式, 能得出这样的结论: dbAcc 的接口协议是遵守 DB 的框架协议。

现在考虑框架和结构协议一致性的实例。设  $P_A$  是 DatabaseV2 的结构协议,  $P_F$  是框架 DatabaseV2 的框架协议。必须验证  $P_A$  遵守限定的框架协议  $P_F \circ_A P_A$  建立字符集  $S$  之上, 组成框架中所有接口之上的事件标识。Sprov 组成  $\langle dbSrv - cRole \rangle$  和  $\langle Local: ds \rangle$  的所有事件标识, Sreq 组成  $\langle sRole - dbAcc \rangle$  和  $\langle sRole - dbLog \rangle$  的所有事件标识, Sint 组成  $\langle Local: tr - cRole \rangle$  和  $\langle sRole - Transm. transm: trans \rangle$  的所有事件标识。首先, 要验证 DatabaseV2 的框架协议。

```
! < sRole - dbAcc >. Open ;
( ? < dbSrv - cRole >. Insert
{ ( ! < sRole - dbAcc >. Insert ; ! < sRole -
dbLog >. LogEvent ) * } + ? < dbSrv - cRole >. Delete
{ ( ! < sRole - dbAcc >. Delete ; ! < sRole - dbLog
>. LogEvent ) * } + ? < dbSrv - cRole >. Query
{ ( ! < sRole - dbAcc >. Query ) * } ) * ;
! < sRole - dbAcc >. Close
```

根据定义,  $P_A$  遵守  $P_F$  要满足下列两个条件:

$L(P_A/P_F)/Sprov \vdash L(P_A)/Sprov$  和  $L(P_A/P_F)/Sprov \vdash L(P_A)/Sext \vdash L(P_A)/Sext$ 。

对于条件 1, 能通过比较下面的协议得到验证。

Database 的框架协议:

```
(? <dbSrv - cRole>. Insert + ? <dbSrv - cRole>. Delete
+ ? <dbSrv - cRole>. Query)
```

Database 的结构协议:

```
* (? <dbSrv - cRole>. Insert + ? <dbSrv - cRole>.
Delete + ? <dbSrv - cRole>. Query
+ ? <Local:ds>. SetTrModel +
? <Local:ds>. GetTrModel ) *
```

通过比对验证,可以得出条件1是满足的;同样,为了验证条件2,要通过下面的协议进行验证。在条件2中调整和限定操作符暗示如下协议通过比较具有包含关系。因为结构协议使用的是需求接口/连接<Local:da-dbAcc>和<Local:lg-dbLog>,所以不用比较框架协议,它们满足条件2。因此,DB的结构协议遵循DB的框架协议。

```
! <sRole - dbAcc>. Open ;
(? <dbSrv - cRole>. Insert
| ( ! <sRole - dbAcc>. Insert ; ! <sRole - dbLog>.
LogEvent ) * } + ? <dbSrv - cRole>. Delete
| ( ! <sRole - dbAcc>. Delete ; ! <sRole - dbLog>. Lo-
gEvent ) * } + ? <dbSrv - cRole>. Query
| ( ! <sRole - dbAcc>. Query ) * } ) * ; ! <sRole -
dbAcc>. Close
! <sRole - dbAcc>. Open ;
(? <dbSrv - cRole>. Insert ! <sRole - dbAcc>. Insert ;
! <sRole - dbLog>. LogEvent
| + ? <dbSrv - cRole>. Delete
| ! <sRole - dbAcc>. Delete ; ! <sRole - dbLog>. Lo-
```

(上接第127页)

合和互补。

由于传统分布式技术在安全性、可靠性、可伸缩性等方面具有一定的优势,因此,可以尝试在企业内部网上使用传统分布式技术。同时,根据Web Service的特性。可以考虑在基于WAN和Internet的应用和基于异构平台的应用中使用Web Services。这样就是可以发挥它们各自的长处,使得异构分布式下的应用从各方面得到加强。

## 5 结束语

Web Service同传统的分布式技术并不是对立的。实际上它们是互补的,不存在级别高低的问题。Web Service目前还并不适合核心的业务领域,其组件模型相比传统分布式技术来说是不成熟的。核心的应用领域还是要靠CORBA等来建造。Web Service最弱的一点即缺乏对象操作能力,而这个恰好是CORBA等分

```
gEvent ! + ? <dbSrv - cRole>. Query
| ! <sRole - dbAcc>. Query ! ) * ;
! <sRole - dbAcc>. Close
```

## 5 总结

提出基于行为协议的构件化软件静态测试技术。本技术在构件通信模型的基础之上,应用行为协议对构件的行为进行形式化验证。从而尽可能地避免静态错误,为下一步实现构件化软件的自动测试工具打下基础。

## 参考文献:

- [1] Szyperski C. 构件化软件——超越面向对象编程[M]. 第2版. 王千祥译. 北京:电子工业出版社,2004:84-152.
- [2] Plasil F, Visnovsky S. Behavior protocols for Software Components[J]. IEEE Trans. on SW Eng., 2002, 28(9): 51-59.
- [3] Medvedovic N, Taylor R N. A Classification and Comparison Framework for Software Architecture Description Languages[R]. Irvine: Department of Information and Computer Science, University of California, 1997: 35-40.
- [4] Allen R J. A Formal Approach to Software Architecture[D]. Pittsburgh: School of Computer Science, Carnegie Mellon University, 1997: 83-92.
- [5] Giannakopoulou D, Kramer J, Cheung S C. Analysing the Behaviour of Distributed Systems using Tracta[J]. Journal of Automated Software Engineering, special issue on Automated Analysis of Software, 1999, 6(1): 72-75.

布式技术的强项。因此,研究和发展Web Service的目的并不是取代传统的分布式技术,而是要更好地同传统的分布式技术结合,以求在技术应用中发挥更好更大的作用。

## 参考文献:

- [1] 高璟,汪洪涛,丁颖,等. 基于XML的新一代模型——Web Services[J]. 微机发展, 2004, 14(1): 93-98.
- [2] Lai R. J2EE Platform Web Services[M]. 周斌,等译. 北京:电子工业出版社,2005.
- [3] 郭宏. 分布式对象技术新进展[J]. 微电子世界, 2003, 22(1): 26-29.
- [4] Otte R, Patrick P, Roy M. CORBA 教程——公共对象请求代理体系结构[M]. 北京:清华大学出版社,2001.
- [5] Bloch J. Effective Java Programming Language Guide[M]. 影印版. 北京:中国电力出版社,2004.
- [6] 郑俊辉,许雷. Web Service 与 CORBA 的比较及分析[J]. 西南民族大学学报, 2005, 31(1): 134-137.