基于Bigraph的面向方面动态软件体系结构演化研究

汪 玲[1] 戎 玫[2] 张广泉[1,3] 王 昇[1]
(苏州大学计算机科学与技术学院 苏州 215006)[1] (暨南大学深圳旅游学院 深圳 518053)[2]
(中国科学院计算机科学国家重点实验室 北京 100080)[3]

摘 要 随着网络技术的发展,软件运行环境的开放化和用户需求的多样化,使得人们对软件动态演化能力提出了更高的要求。面向方面软件开发中的关注点分离思想很好地支持了软件动态演化,现有的形式化方法难以直观地表示体系结构的动态性,且不能很好地验证系统演化前后的正确性。Bigraph不仅具有直观的图形化表达能力,而且具备良好的数学基础,可以推理和验证系统的演化性质。因此,提出了一种面向方面动态软件体系结构(AODSA)模型,扩展的Bigraph用于描述AODSA的结构,使用Bigraph反应系统(BRS)来描述AODSA的动态演化。最后以一个简化的ATM存款系统模型为例,说明Bigraph表示的ATM模型以及基于BRS的演化过程。

关键词 Bigraph,Bigraph反应系统,AODSA,动态演化

Research on Bigraph-based Aspect-oriented Dynamic Software Architecture Evolution

WANG Ling[1] RONG Mei[2] ZHANG Guang-quan[1,3] WANG Sheng[1]
(School of Computer Science and Technology, Soochow University, Suzhou 215006, China)[1]
(Tourism College, Jinan University, Shenzhen 518053, China)[2]
(The State Key Laboratory of Computer Sciences, Chinese Academy of Sciences, Beijing 100080, China)[3]

Abstract As the development of network technology, the runtime environment of software is becoming more and more complicated and the requirements of software users are growing with diversification. These variations lead to more advanced demands on the ability of dynamic evolution of software. The concept, which is called separation of concerns, during the development of aspect-oriented software development, can well support the dynamic evolution of software. The existing formal methods can not represent the dynamicity of software architecture intuitively; however, more important is that they can not efficiently verify the validity of the system before and after the evolution. Bigraph not only has the ability of intuitive graphical representation, but also possesses better mathematical foundation. Therefore, we proposed a new model, which is called Aspect-Oriented Dynamic Software Architecture (AODSA), to solve these problems. First, Bigraph was extended in order to describe the structure of AODSA. Then, Bigraph reactive system (BSR) was used to represent the dynamic evolution of AODSA. At last, a simple ATM deposit system model was used for example to illustrate the usage of AODSA.

Keywords Bigraph, Bigraph reactive system, AODSA, Dynamic evolution

1 引言

近年来,随着互联网的飞速发展,计算机软件面临的环境从静态封闭逐步走向动态开放。为了适应这种变化趋势,人们期望软件系统在执行过程中能感知外部环境的动态变化,并在不暂停系统运行的情况下,随着这种变化按照功能指标、性能指标和可靠性指标等进行动态演化。因此,如何建立一个可以支持软件动态演化的软件系统模型,并且保证系统动态演化前后的正确性,成为一个新的挑战。

在传统的面向对象软件开发中,当多个互不相连的模块之间有一些相同的非功能模块发生演化时,每个模块中涉及到演化的非功能模块都要进行修改,这就增加了系统演化的复杂度,从而导致代码缠结。面向方面编程(Aspect-Oriented Programming, AOP)[1]的提出很好地解决了上述问题。它运用关注点分离思想,将这些共同的非功能模块横切出来,模块化为方面(aspect)。当包含方面的多个模块发生演化时,只需要对方面进行修改,然后使用编织技术将修改后的方面编织到需要演化的模块中,从而实现了系统的动态演化并解决了代码缠结问题。然而,AOP只是一种编程技术,不容易实现方面的复用。因此,本文从体系结构层次和支持软件动态演

到稿日期:2009-10-26 返修日期:2009-12-29 本文受中国科学院计算机科学国家重点实验室开放课题(SYSKF0908),江苏省高校自然科学研究项目(08KJB520010)资助。

汪 玲(1985—),女,硕士生,主要研究方向为面向方面软件体系结构、形式化方法,E-mail:gqzhang@suda.edu.cn;戎 玫(1966—),博士,副教授,CCF高级会员,主要研究方向为形式化方法、软件工程与电子商务;张广泉(1965—),男,博士,教授,CCF高级会员,主要研究方向为软件工程与形式化方法、可信计算等;王 昇(1983—),男,硕士生,主要研究方向为面向服务架构、形式化方法。

化的角度出发，通过建立一种面向方面动态软件体系结构模型（Aspect-Oriented Dynamic Software Architecture，AODSA），来研究体系结构的动态演化。为了能够分析和描述系统的动态性，目前大多数学者都使用动态体系结构描述语言对软件体系结构的动态性进行描述。虽然这种方法能够提供准确的表示方法来描述系统的动态性，但是其描述都过于符号化，缺乏直观的表达能力，且难以对系统演化的性质进行推理和验证。Bigraph 是一种基于图形的形式化工具[2]，它融合了 π 演算和移动 Ambient 演算的优势，不但能够直观地表示体系结构的拓扑结构，而且 Bigraph 反应系统（BRS）还可以对体系结构的动态演化进行描述。此外，Bigraph 具备的完整的公理系统可以对动态软件体系结构的演化性质进行分析和验证。

本文第 2 节简单介绍 Bigraph 的基本概念；第 3 节建立面向方面动态软件体系结构模型（AODSA）；第 4 节建立 AODSA 的 Bigraph 模型，使用 Bigraph 反应系统表示 AODSA 的演化过程，并以 ATM 存款系统为例来说明 AODSA 的建模和演化过程；第 5 节是相关工作比较；最后总结全文并提出下一步工作。

2 Bigraph 简介

Bigraph 是由图灵奖获得者 Robin Milner 和他的合作者于 2001 年提出的一种基于图形的形式化工具，它是在进程计算、移动 Ambient 演算以及 π 演算的基础上演化而来的，主要用来为普适计算系统以及移动和并发理论建模[2]。下面从静态结构和动态结构两个不同的角度对 Bigraph 进行阐述。从静态结构方面来说，一个 Bigraph 可以分解为两个子图：位置图和连接图[3]。

定义 1 Bigraph $G=(V,E,Ctrl,GP,GL):I\rightarrow J$，$I=\langle m,X\rangle$，$J=\langle n,Y\rangle$。其中，$V$：有限的节点集。$E$：有限的边集合。$Ctrl:V\rightarrow K$，节点到控制的映射图，表示每一个节点 V 都有一个对应的控制 K，K 用来描述该节点的端口，并规定该节点是原子节点还是复合节点。GP：位置图，表示各计算节点的所在位置，节点之间可以相互嵌套。GL：连接图，表示节点间的连接关系。I,J 分别为内部接口和外部接口。

定义 2（Bigraph 位置图） $GP=(V,Ctrl,Prnt):m\rightarrow n$。其中，$m$：内部位置的个数。$n$：外部区域的个数。$Ctrl:V\rightarrow K$，节点的控制图。双亲图 $Prnt:m\cup v\rightarrow v\cup n$，双亲图是一个非循环图，它是由 n 个树组成的森林，外部区域名为每个树的根节点。

定义 3（Bigraph 连接图） $GL=(V,E,Ctrl,Link):X\rightarrow Y$。其中，$X$：连接图的内部名。$Y$：连接图的外部名。$Ctrl:V\rightarrow K$，控制图。$Link:X\cup P\rightarrow E\cup Y$，其中 P 是 G 中所有节点的接口。

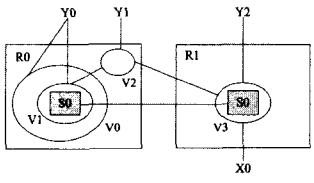

图 1 Bigraph G

图 1 所示即是一个简单的 Bigraph G，$G=(V,E,Ctrl,GP,GL):\langle 1,X\rangle\rightarrow\langle 2,Y\rangle$，$X=\{X0\}$，$Y=\{Y0,Y1,Y2\}$。图中有两个位置 $S0,S1$，有两个区域 $R0,R1$。图 2 和图 3 分别表示的是 G 的位置图和连接图。

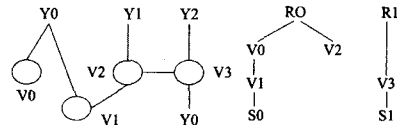

图 2 G 的连接图　　图 3 G 的位置图

Bigraph 不仅具有直观的图形化表达方式以方便建模，而且还有相关的项语言（term language）对系统性质进行推理和演绎。基本项语言元素如表 1 所列[5]。

表 1 基本项语言元素

U||V	并列根区域
U| V	并列一个根区域下的子节点
U∘V	组合操作
U⊗V	扩展乘积
U(V)	包含，U 节点包含 V 节点
Kx	控制 K 有外部名 X
1	空的区域
−i	地点名为 i
/x U	U 的外部名 X 被一条边代替
x/y	连接内部名 y 和外部名 x

从动态性方面来说，Bigraph 和一组反应规则可组成 Bigraph 反应系统[4]，并通过反应规则对自身进行重配置，可以从一个 Bigraph 演化到另外一个 Bigraph。

定义 4 Bigraph 反应系统 $S=\langle R,r,R'\rangle:R\rightarrow R'$。$R$ 和 R' 分别为反应物和生成物，都是 Bigraph，r 为反应规则并规定了 R 到 R' 的反应过程。图 4 就是一个 Bigraph 反应系统。

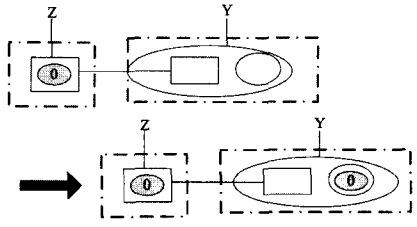

图 4 Bigraph 反应系统

3 面向方面动态软件体系结构模型（AODSA）

面向方面编程的关键技术就是识别和分离关注点，软件系统可以看成关注点的集合。根据关注点的性质不同可分为核心关注点和横切关注点。核心关注点是软件要实现的主要功能和目标；横切关注点是那些与核心关注点之间有横切作用的关注点，一般是系统的一些非功能属性。已有的体系结构元素构件、连接件无法表达 AOP 中的横切关注点。因此，使用一种新的体系结构元素——方面来封装横切关注点。建立面向方面的动态体系结构模型，需要一个专门的机制来实现和管理 AODSA 的动态演化。本文对已有的体系结构配置进行扩展，将配置分为基础配置和动态重配置。其中基础配置是关于体系结构拓扑结构的描述，而动态重配置管理 AODSA 的动态演化。AODSA 的核心模型如图 5 所示。

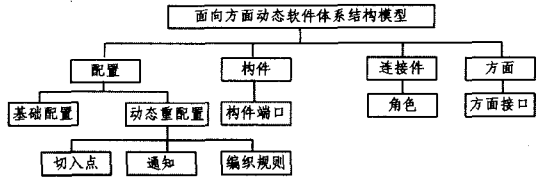

图 5 面向方面动态软件体系结构的核心模型

图中，构件(component)：封装核心关注点功能模块。

方面(aspect)：封装横切关注点功能模块。

动态重配置(dynamic configuration)：管理 AODSA 的动态演化，由切入点、通知和编织规则组成。

切入点(pointcut)：是一个二元组〈构件的接口，方面的接口〉，定义了需要演化到构件中的方面接口和演化构件的端口。

通知(advice)：封装定义了什么时候调用方面(之前、之后、替代)，也就是在切入点中定义的构件接口之前(之后，还是替代)调用方面的接口。

编织规则(weaving rule)：它是个三元组〈方面，通知，切入点〉，描述了在每次调用切入点定义的构件接口之前(之后，替代)将调用切入点定义的方面接口。

4 面向方面动态软件体系结构的 Bigraph 模型

本节用 Bigraph 描述面向方面动态软件体系结构，主要描述 AODSA 的结构和动态演化操作。首先对已有的 Bigraph 进行扩展，使得它能够表示 AODSA 的静态结构。然后运用 BRS 中的反应规则表示 AODSA 的动态演化操作，将 AODSA 上的演化操作转换为 BRS 上的操作。

4.1 AODSA 的 Bigraph 模型

已有的 Bigraph 无法描述 AODSA 的结构。因此，我们将 Bigraph 进行扩展，使得它能够描述 AODSA 的结构。

定义 5 节点的控制 K 是一个三元组〈port name，port attribute，point type〉。其中，port name(端口名)：若节点表示构件，则为构件接口名；若表示连接件，则为连接件角色名；若表示方面，则为方面的端口名。port attribute(端口的属性)：定义了构件、连接件或者方面的功能属性。point type(节点的类型)：一个布尔变量，"TRUE"表示该节点是个复合节点，即它可以发生演化；"FALSE"表示该节点是一个原子节点，不能发生演化。

定义 6 扩展的 Bigraph $F=(Q,C,A,E,CtrlQ,CtrlC,CtrlA,FL,FP):I\rightarrow J$。其中，

Q：表示构件节点。

C：表示连接件节点。

A：表示方面节点。

E：一组连接各个接口的边。

$CtrlQ$：构件的控制，用来管理构件相关信息。

$CtrlC$：连接件的控制，用来管理连接件相关信息。

$CtrlA$：方面的控制，用来管理方面相关信息。

FL：扩展的连接图。

FP：扩展的位置图。

I,J：构件、方面或者连接件的接口。

下面以一个 ATM 系统为例来说明。在这个 ATM 系统中，我们只考虑它的自动取款功能，其核心关注点是存款模块和用户登录模块。横切关注点主要有身份验证方面，在用户登录和进行存款的时候都需要对用户身份进行验证，所以将身份验证模块化方面。图 6 是用 Bigraph 表示的 ATM 系统的面向方面体系结构模型 BATM。

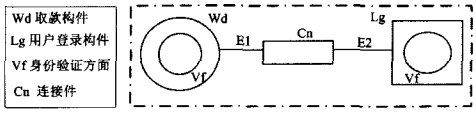

图 6 ATM 系统的 Bigraph 模型 BATM

该 Bigraph BATM=({Wd,Lg},Cn,Vf,{E1,E2},{WD,LG},CN,VF,BATML,BATMP)。其中，节点 Wd，Lg 表示构件，Cn 表示连接件，Wd 和 Lg 中嵌套的节点 Vf 用来表示横切关注点身份验证方面，E1 和 E2 表示构件和连接件之间的交互。Wd，Lg，Vf，Cn 的控制分别为 WD{port withdraw，attribute withdraw，TRUE}，LG{port login，attribute login，TRUE}，VF{port verification，attribute verification，TURE}，CN{(port login1，withdraw1)，(attribute login1，withdraw1)，FALSE}。控制 WD，LG 分别表示了构件 Wd，Lg 是可演化的，以及它的接口名、接口的属性等。VF 表示方面 Vf 是可演化的，以及它的接口名和属性。CN 表示连接件不可演化并指出它的接口信息。

用项语言形式化表示图 6 中的 Bigraph 模型如下：$BATM=/x./y.^{x}/_{withdraw,withdraw1}.^{y}/_{login,login1}\ \ Wd._{withdraw}(Vf._{verfication})\mid Cn._{withdraw1.login1}\mid Lg._{login}(Vf._{verfication})$。

4.2 用 BRS 表示 AODSA 的动态演化

为了能管理和实现 AODSA 的动态演化，在 AODSA 模型中加入了动态重配置机制。BRS 中的反应规则可以表示动态重配置，则 AODSA 的动态演化过程就可以映射为 BRS 中一个 Bigraph 通过反应规则转换为另外一个 Bigraph 的过程。在 BRS $S=\langle R,r,R'\rangle$ 中，R 和 R' 分别表示演化前和演化后的 AODSA 模型，r 表示体系结构发生动态演化的操作。

为了能突出面向方面体系结构演化的特点，本文只关注 AODSA 中方面的演化。方面的演化主要包括 3 种类型：增加方面、删除方面和修改方面。

增加方面(add aspect)：当系统中一个或者多个模块需要增加一些相同的非功能模块时，可将这些非功能模块看作方面，并通过增加方面操作来实现演化。若该方面模块已经存在，则只需通过编织方面操作 weaving(aspect，pointcut，advice，component)将方面模块编织到需要演化的模块中；若方面模块之前没有出现过，则需要先通过创建方面操作 create aspect()建立一个新的方面，然后再将它编织到需要演化的模块中。

删除方面(delete aspect)：当系统的一个或者多个模块中某些非功能模块不再受到关注时，可通过删除这些非功能模块(方面)来实现。主要用到的操作有 detach(aspect，component)，通过断开方面接口和需要演化的模块的端口来实现软件演化。

修改方面(modify aspect)：当系统的一个或者多个模块中相同的非功能属性需要升级或者修改时，可通过修改方面来实现。该方法可通过组合删除方面、修改方面和编织方面操作来实现。修改方面的操作可分为对方面功能修改和接口修改，以达到用户需求。

weaving(aspect，pointcut，advice，component)作为方面演化过程中的重要操作，其具体实现过程就是调用 pointcut 中指定的构件接口之前、之后、替代(由 advice 决定的)调用 aspect 的端口。

将 AODSA 看成一个 Bigraph，对体系结构方面的演化操作可通过对 Bigraph 进行转换来实现。首先需要将上述的演化操作转换为用 Bigraph 的项语言表示的 Bigraph 图形转化操作。

create aspect(A) = A.$_{aport}$//建立新的方面 A 及端口 aport

weaving aspect(A, pointcut, advice, C) = /x.^{x}/$_{aport,cport}$. C.$_{cport}$(A.$_{aport}$) // aport,cport 为切点中指定的方面接口和构件端口。

detach aspect(A,C) = $^{\varepsilon}$/$_{aport,cport}$. C.$_{cport}$(A.$_{aport}$) // 断开方面 A 和构件 C 的接口连接。

在 ATM 取款系统中,用户在登录系统以及取款时都需要对身份进行验证。为了保证客户信息的安全,当用户取款时,需要再次对用户的取款权限进行验证,因此在取款构件执行之前加入验证用户取款权限方面(Au),这就需要对系统进行演化。演化操作如下:

```
add aspect {
create aspect 取款权限()
weaving(取款权限,(取款权限接口,存款端口),之前,存款构件)
}
```

我们也可以用 BRS 中的反应规则来直观地表达这种演化操作,如图 7 所示,同时用项语言来形式化地表示这种演化过程。

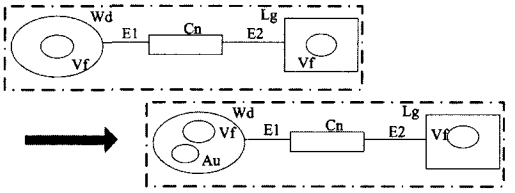

图 7 ATM 演化图的 BRS 表示

转换后的 Bigraph 为 BATM′,则它的项语言表达式为 BATM′ = /x. /y. ^{x}/$_{withdraw,withdraw1}$. ^{y}/$_{login,login1}$. Wd.$_{withdraw}$(Vf.$_{verfication}$ | Au.$_{authority}$) | Cn.$_{withdraw1,login1}$ | Lg.$_{login}$(Vf.$_{verfication}$)。

如图 7 所示,在 BRS 中使用反应规则,在 Bigraph BATM 中的 Wd 节点中嵌套加入新的方面节点 Au,使其转换为 Bigraph BATM′,实现了在 ATM 系统中用户取款之前先对用户的取款权限进行验证,从而更好地保障了用户的信息和财产的安全。

5 相关工作

目前,已有一些学者对面向方面动态软件体系结构的建模和描述做了一些研究工作。面向方面动态体系结构建模主要从两个不同的角度出发:一种是对称式建模方法,即将系统所有的功能模块和非功能模块都封装为方面,通过编织这些方面构成软件体系结构中的构件和连接件。如 Jennifer Perez 等人提出了一种对称式体系结构模型 PRISMA,该模型中的首要元素是方面,构件、连接件都是由方面编织而成的,通过配置方面来管理系统的动态演化[8-10]。另外一种是非对称式建模方法,即将系统的横切模块封装为方面,功能模块封装为构件。系统的主要元素依然是构件和连接件,方面只是作为一种特殊的元素,通过将方面无缝地编织到构件中,从而达到一种非侵入式的演化方法。复旦大学的赵文耘等通过在增量 AO 体系结构模型中建立一个特殊连接件模型——方面编织连接件(AWC)来支持系统的动态演化[11]。而对动态体系结构的形式化描述来说,一般都使用体系结构描述语言,如 PRISMA ADLG、扩展的 ACME ADL 来对系结构模型以及演化过程进行描述和分析。

Bigraph 融合了 π 演算和移动 Ambient 演算的优势,是一种基于图形的形式化工具。以 Milner 为代表的学者已经对 Bigraph 做了一些研究工作[1-3,5],同时有学者用 Bigraph 描述上下文感知系统[7]。丹麦哥本哈根信息工程大学正在开发基于 Bigraph 的普适计算语言(BPL, Bigraphical Programming Language)[4],该语言很有可能替代现有的 UML 等建模语言,成为普适计算的主流建模语言。国内也有学者用 Bigraph 理论对自适应软件体系结构进行描述和验证[6]。本文将 Bigraph 应用到面向方面动态软件体系结构中,还属于首次尝试。

结束语 本文从支持软件动态演化的角度出发,在基于构件开发模型的基础上,将横切关注点封装为方面,建立了一个面向方面动态软件体系结构模型(AODSA),并用动态重配置机制来管理系统的动态演化。扩展了 Bigraph 并对该模型以及它动态演化的性质进行描述和分析,首先使用扩展的 Bigraph 对 AODSA 静态结构进行建模,然后运用 BRS 中的反应规则来描述 AODSA 的动态演化操作,使得对 AODSA 上的演化操作转换为对 BRS 上的 Bigraph 转换操作,即一个 Bigraph 通过反应规则转换为另外一个 Bigraph。下一步工作主要是利用 Bigraph 良好的数学基础来推理和验证 AODSA 动态演化的性质。

参 考 文 献

[1] Kiczales G, Lamping J, Mendhekap A, et al. Aspect-oriented programming[C]// Proceedings of ECOOP'97. Finland: Springer-Verlag, 1997: 220-242

[2] Milner R. Bigraphical reactive systems: basic theory [R]. UCAM-CL-TR-523University of Cambridge, 2001

[3] Jensen O-H, Milner R. Bigraphs and mobile processes(revised) [R]. UCAM-CL-TR-580. University of Cambridge, 2003

[4] Birkedal L, Bundgaard M, Milner R, et al. Bigraphical Programming Languages for Pervasive Computing[C]// Proceedings of Pervasive International Workshop on Combining Theory and Systems Building in Pervasive Computing. 2006: 653-658

[5] Milner R. Axioms for bigraphical structure[J]. Journal of Mathematical Structures in Computer Science, 2005, 15(6): 1005-1032

[6] 常志明,毛新军,齐治昌. Bigraph 理论在自适应软件体系结构上的应用[J]. 计算机学报, 2009, 32(1): 97-106

[7] Birkedal L, Debois S, Elsborg E, et al. Bigraphical Models of Context-aware Systems[J]. Journal of Mathematical Structures in Computer Science, 2005, 15(6): 1005-1032

[8] Perez J, Ramos I, Jaen J, et al. PRISMA: Towards Quality, Aspect Oriented and Dynamic Software Architectures[C]// Proceedings of the 3rd International Conference on Quality Software(QSIC'03). Los Alamitos, CA: IEEE CS Press, 2003: 59-66

[9] Perez J, Ali N, Carsi J A, et al. Dynamic Evolution in Aspect-oriented Architectural Models[C]// Proceedings of EWSA'05. LNCS 3527. Berlin: Springer-Verlag, 2005: 59-76

[10] Costa C, Ali N, Perez J, et al. Dynamic Reconfiguration of Software Architectures Through Aspects[C]// Proceedings of ECSA'07. LNCS 4758. Berlin: Springer-Verlag, 2007: 279-283

[11] Lau Yi-Ming, Zhao Wen-Yun, Peng Xin, et al. A unified formal model for supporting aspect-oriented dynamic software architecture[C]// Proceedings of ICCIT'07. Los Alamitos, CA: IEEE CS Press, 2007: 450-455