软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(10):2735−2745 [doi: 10.3724/SP.J.1001.2012.04170]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

基于构件运算的可重构系统代数模型*

袁 博+, 汪斌强

(国家数字交换系统工程技术研究中心,河南 郑州 450002)

Algebraic Model of Reconfigurable System Based on Component Operation

YUAN Bo+, WANG Bin-Qiang

(National Digital Switching System Engineering and Technology Center, Zhengzhou 450002, China)

+ Corresponding author: E-mail: yuanbonet@163.com

Yuan B, Wang BQ. Algebraic model of reconfigurable system based on component operation. *Journal of Software*, 2012,23(10):2735−2745 (in Chinese). http://www.jos.org.cn/1000-9825/4170.htm

Abstract: Reconfigurable systems, composed by many components, can exhibit different function caused by component combinations and alternations. This paper focuses on the lack of methodology used in describing formal model and reconfigurable attributes of the reconfigurable system. The paper also focuses on the abstracts and describes the attributes and the behaviors of reconfigurable system and the component combination and reconfigurable method by an algebraic method. By understanding the component combination of an operation that is a new idea and extending the calculus in algebraic process, some component combination operators are defined. Next, a formal algebraic model of reconfigurable system is proposed. Based on this model, some reconfigurable attributes are analyzed and a few reconfigurable nomal formats are proposed. All above viewpoints construct the theoretical footstone for designing reconfigurable system. Finally, a case study is introduced to explain how the above algebraic model can be used.

Key words: component; component combination; reconfigurable system; process algebraic; reconfiguration modeling

摘 要: 可重构系统是指一个系统由构件组成,随着构件被替换以及组合拓扑关系的变化,系统表现出不同的功能.针对可重构系统在形式化和重构建模方面的不足,用代数学方法对可重构构件、构件组合、可重构系统的属性和行为特征进行抽象,把构件组合定义成构件的“运算”实现,结合进程代数中算子的概念,定义了多种构件组合运算,建立了可重构系统的代数模型.在代数模型基础上,提出了重构建模和重构范式,为可重构系统提供理论支持,最后介绍了应用案例.

关键词: 构件;构件组合;可重构系统;进程代数;重构建模

中图法分类号: TP311 **文献标识码**: A

基于构件的软件工程(component based software engineering,简称 CBSE)通过组装可复用、可插拔的软件构件来构造软件系统.CBSE 可以看作是从面向对象技术和软件构件技术演化而来的一种新的体系结构风格,从当前的学术研究和工业实践来看,许多针对 CBSE 的研究和技术来源正是构件技术的研究成果.构件的一个优

* 基金项目: 国家高技术研究发展计划(863)(2008AA01A323, 2009AA01A334); 国家重点基础研究发展计划(973)(2012CB315900); 国家科技支撑计划(2011BAH19B01)

收稿时间: 2010-12-12; 修改时间: 2011-09-22; 定稿时间: 2011-12-31

良特性便是构件间拓扑关系可重构,本文从可重构的角度出发解决面向服务的软件开发问题,用代数学方法对重构、构件组合、可重构系统的属性和行为特征进行抽象,把构件组合定义成构件的“运算”实现,结合进程代数中算子的概念,定义了多种构件组合运算,建立了可重构系统的代数模型,为可重构系统的设计和实现提供理论支持.

可重构系统可被定义成为适应需求的变化,以重新排列、重复利用、更新构件的方式使系统功能发生变化的一类可动态组织的系统.为了使系统具有重构能力,一方面,系统必须具有较强的演化能力和适应性,支持快速、高效的重构;另一方面,系统应具有良好的体系结构,重构前后的体系结构变化不应太大.系统必须是柔性的,这种柔性是指系统具有一种弹性的可以调整的组织结构,可以兼容某一类应用,主要体现在两个方面:一是系统级重构,即系统可动态地增加或减少构件组合;二是单元级重构,可增加构件内部的领域知识.

本文从软件体系结构建模入手,用进程代数对可重构系统的建模方法进行研究,通过对传统进程代数的扩展,提出多种服务组合运算,建立可重构系统的代数模型;并在此基础上研究可重构系统的设计问题,为可重构系统设计提供理论支持.本文第 1 节给出可重构系统代数模型描述.第 2 节介绍如何针对可重构系统的可重构属性进行建模.第 3 节对 3 个应用案例进行分析.第 4 节介绍相关研究工作.最后总结全文.

1 可重构系统代数模型

本节解释可重构构件、构件组合以及可重构体系结构等概念,最后给出可重构系统代数模型.

1.1 可重构构件

我们将可重构系统理解为:“具有服务能力的、可按照一定的策略和特性(如拓扑关系)组合的构件集合.”下面分别给出可重构构件、构件组合以及可重构体系结构的形式化描述以及服务重构概念.

定义 1(可重构构件(简称构件)). 可重构构件是具有相对独立功能、可以明确辨识、接口由契约指定、可独立部署、和语境有明显依赖关系、且多由第三方提供的可组合软件实体.构件由构件接口和构件处理模块组成.构件是一个$\langle ID,Port,Deal,Fun\rangle$四元组,其中,$ID$ 代表构件的标识;$Port$ 是构件接口,表示构件与外部接触点的集合;$Deal$ 是处理模块,表示构件内部处理功能集合;Fun 描述构件的功能,包括构件的服务策略、安全性、可靠性等.其中,$Port$ 集合的每个触点由一个五元组$\langle PoID,Extn,Publ,Msg,Cons\rangle$表示.例如,对第 i 个触点,可用$Port_i=\langle PoID,Extn_i,Publ_i, Msg_i,Cons_i\rangle$表示,其中,

- $PoID$ 是构件端口标识;
- $Extn_i$ 表示第 i 个触点从外界环境获得的功能(操作),操作执行记为事件 r;
- $Publ_i$ 表示第 i 个触点提供给外界环境的功能(操作),操作执行记为事件 s;
- Msg_i 是触点消息的集合,$Msg(r)$表示事件 r 从外界接收的消息,$Msg(s)$表示事件 s 提供给外界的消息;
- $Cons_i$ 是对第 i 个触点的行为约束,用 $Cons(init,per\text{-}cond,post\text{-}cond)$表示,$init,per\text{-}cond,post\text{-}cond$ 表示触点行为的初始条件、前置条件和后置条件.

下面给出构件相等的概念.

定义 2. 设 A,B 是论域 $Dom(U)$中的两个构件,如果 A,B 满足下列条件:

$$\begin{cases} Deal(A)=Deal(B) \\ Port(A)=Port(B) \\ Deal(A)=Deal(B) \\ Fun(A)\Leftrightarrow Fun(B) \\ Extn(P_A)=Extn(P_B) \\ Publ(P_A)=Publ(P_B) \\ Msg(s_A)=Msg(s_B) \\ Msg(r_A)=Msg(r_B) \\ Cons(P_A)\Leftrightarrow Cons(P_B) \end{cases} \tag{1}$$

则称 A,B 相等,记作 $A=B$.其中,⇔表示逻辑等价,下同.

1.2 构件组合

可重构系统的特点是构件可以复用,并且可以通过构件连接关系的变化引起系统功能的变化.为此,我们结合构件的特点对进程代数中的算子进行扩展,把构件组合解释成构件之间的连接运算的实现,为构件间的连接方式提供了一种形式化描述方法.

定义 3. 设构件 A,B 是论域 $Dom(U)$中的两个构件,若$\exists s\in Publ(P_A)\wedge\exists r\in Extn(P_B)$使得$[per\text{-}cond(P_A)\wedge per\text{-}cond(P_B)\wedge(s\Rightarrow r)]\wedge[Msg(s)=Msg(r)]$,即构件 A 通过发送一个消息"激发"构件 B 来实现功能需求,则称构件 A,B 进行了一次单向"激发"运算,记作 $A\vec{\oplus}B$.其中,P_A 和 P_B 表示构件 A 和构件 B 的接口,下同.

定义 4. 设构件 A,B 是论域 $Dom(U)$中的两个构件,若$\exists r\in Extn(P_A)$,对$\forall s\in Publ(P_B)$,若$[(s\Rightarrow r)\wedge(per\text{-}cond(P_A)\wedge per\text{-}cond(P_B))]\wedge[Msg(s)\in Msg(r)]$,则称构件 A 与构件 B 进行了一次带有"反馈"的"激发"运算,即 A 与 B 是双向激发,记作 $A\vec{\oplus}B$.$A\vec{\oplus}B$ 和 $A\vec{\oplus}B$ 记为 $A\oplus B$,称作"激发"运算.特别地,当 $B=\varnothing$时,$A\oplus B=A$.

如果把 $A\oplus B$ 看成是一个整体,则 $A\oplus B$ 仍是一个构件,记作 $A\Delta B$,并且满足下列性质:

$$\begin{cases} Dom(A\Delta B)=Dom(A)\cup Dom(B)\\ Deal(A\Delta B)=Deal(A)\cup Deal(B)\\ Port(A\Delta B)=Port(A)\cup Port(B)\\ Extn(P_{A\Delta B})=Extn(P_A)\\ Publ(P_{A\Delta B})=Publ(P_B)\\ Msg(s_{A\Delta B})=Msg(s_A)\cup Msg(s_B)\\ Msg(r_{A\Delta B})=Msg(r_A)\cup Msg(r_B)\\ Cons(P_{A\Delta B})=Cons(P_A)\cup Cons(P_B) \end{cases} \tag{2}$$

定义 5. 设构件 A,B 是论域 $Dom(U)$中的两个构件,若$\exists s\in Publ(P_A)\wedge\exists r\in Extn(P_A)\wedge\exists m\in Extn(P_B)$使得$[per\text{-}cond(P_A)\wedge per\text{-}cond(P_B)\wedge(s\Rightarrow m)]\wedge[r=Deal(B)]$,即构件 A 通过"调用"构件 B 来实现功能需求,则称构件 A,B 进行了一次"调用"运算,记作 $A\otimes B$.显然,"调用"运算必然包含"反馈"运算.

如果把 $A\otimes B$ 看成是一个整体,则 $A\otimes B$ 仍是一个构件,记作 $A\nabla B$,并且满足下列性质:

$$\begin{cases} Dom(A\nabla B)=Dom(A)\cup Dom(B)\\ Deal(A\nabla B)=Deal(A)\cup Deal(B)\\ Port(A\nabla B)=Port(A)\\ Extn(P_{A\nabla B})=Extn(P_A)\\ Publ(P_{A\nabla B})=Publ(P_A)\\ Msg(s_{A\nabla B})=Msg(s_A)\\ Msg(r_{A\nabla B})=Msg(r_A)\\ Cons(P_{A\nabla B})=Cons(P_A) \end{cases} \tag{3}$$

"激发"和"调用"运算是最基本的构件组合用算,在不需要区分二者使用环境条件下,可将二者统称为"使用"运算,记作 $A\cup B$.

定义 6. 设构件 A,B 是论域 $Dom(U)$中的两个构件,若$\exists s\in Publ(P_A)$,$\exists r\in Extn(P_B)$可以有$[(per\text{-}cond(B)\wedge r)\Rightarrow(per\text{-}cond(A)\wedge s)]$成立,反之$\exists r\in Extn(P_A)$,$\exists s\in Publ(P_B)$可以有$[(per\text{-}cond(A)\wedge s)\Rightarrow(per\text{-}cond(B)\wedge r)]$成立,但是在运算过程中,$per\text{-}cond(A)\cap per\text{-}cond(B)=\varnothing$,则称构件 A 与构件 B 并行,记作 $A\|B$.

以上定义了两个构件间的连接运算情况,接下来讨论 3 个构件间的运算情况.

定义 7. 设构件 A,B,C 是论域 $Dom(U)$中的 3 个构件,若$\exists r_A\in Extn(P_A)\wedge\exists r_B\in Extn(P_B)\wedge\exists s_{CA},s_{CB}\in Publ(P_C)$,使得$[(per\text{-}cond(P_C)\wedge per\text{-}cond(P_A))\vee(per\text{-}cond(P_C)\wedge per\text{-}cond(P_B))\wedge(s\Rightarrow r)]\wedge[Msg(s_{CA},s_{CB})\rightarrow Msg(r_A,r_B)]$,即构件 C 通过发送消息异步地"激发"构件 A 和构件 B 来实现功能需求,则称 C 与 A,B 进行了一次异步"激发"运算,记作 $C\updownarrow(A\|B)$.

如果把 $C\updownarrow(A\|B)$看成一个整体,则 $C\updownarrow(A\|B)$仍是一个构件,并且满足下列性质:

$$\begin{cases} Dom(C\updownarrow(A\|B))=Dom(A)\cup Dom(B)\cup Dom(C)\\ Deal(C\updownarrow(A\|B))=Deal(A)\cup Deal(B)\cup Deal(C)\\ Port(C\updownarrow(A\|B))=\overline{Port(A)\cap Port(B)\cap Port(C)}\\ Extn(Port_{C\updownarrow(A\|B)})=Extn(P_C)\\ Publ(Port_{C\updownarrow(A\|B)})=Publ(P_A)\cup Publ(P_B)\\ Msg(s_{C\updownarrow(A\|B)})=Msg(s_A)\cup Msg(s_B)\\ Msg(r_{C\updownarrow(A\|B)})=Msg(r_C)\\ Cons(Port_{C\updownarrow(A\|B)})=Cons(P_A)\cup Cons(P_B)\cup Cons(P_C) \end{cases} \tag{4}$$

定义 8. 设构件 A,B,C 是论域 $Dom(U)$中的 3 个构件,若$\exists r_A\in Extn(P_A)\wedge\exists r_B\in Extn(P_B)\wedge\exists s_C\in Publ(P_C)$,使得$[(per\text{-}cond(P_C)\wedge per\text{-}cond(P_A))\wedge(per\text{-}cond(P_C)\wedge per\text{-}cond(P_B))\wedge(s\Rightarrow r)]\wedge[Msg(s_C)\rightarrow Msg(r_A,r_B)]$,即构件 C 通过发送消息同步地“激发”构件 A 和构件 B 来实现功能需求,则称 C 与 A,B 进行了一次同步“激发”运算,记作 $C\uparrow(A\|B)$.

如果把 $C\uparrow(A\|B)$看成一个整体,则 $C\uparrow(A\|B)$仍是一个构件,并且满足下列性质:

$$\begin{cases} Dom(C\uparrow(A\|B))=Dom(A)\cup Dom(B)\cup Dom(C)\\ Deal(C\uparrow(A\|B))=Deal(A)\cup Deal(B)\cup Deal(C)\\ Port(C\uparrow(A\|B))=\overline{Port(A)\cap Port(B)\cap Port(C)}\\ Extn(Port_{C\uparrow(A\|B))}=Extn(P_C)\\ Publ(Port_{C\uparrow(A\|B)})=Publ(P_A)\cup Publ(P_B)\\ Msg(s_{C\uparrow(A\|B)})=Msg(s_A)\cup Msg(s_B)\\ Msg(r_{C\uparrow(A\|B)})=Msg(r_C)\\ Cons(Port_{C\uparrow(A\|B)})=Cons(P_A)\cup Cons(P_B)\cup Cons(P_C) \end{cases} \tag{5}$$

定义 9. 设构件 A,B,C 是论域 $Dom(U)$中的 3 个构件,若$\exists s_A\in Publ(P_A)\wedge\exists s_B\in Publ(P_B)\wedge\exists r_{CA},r_{CB}\in Extn(P_C)$,使得$[(per\text{-}cond(P_A)\wedge per\text{-}cond(P_C))\vee(per\text{-}cond(P_B)\wedge per\text{-}cond(P_C))\wedge(s\Rightarrow r)]\wedge[Msg(s_A,s_B)\rightarrow Msg(r_{CA},r_{CB})]$,即构件 A、构件 B 通过发送消息“非互斥”地“激发”构件 C 来实现功能需求,则称 A,B 与 C 进行了一次非互斥“激发”运算,记作$(A\|B)\nearrow C$.

如果把$(A\|B)\nearrow C$ 看成一个整体,则$(A\|B)\nearrow C$ 仍是一个构件,并且满足下列性质:

$$\begin{cases} Dom((A\|B)\nearrow C)=Dom(A)\cup Dom(B)\cup Dom(C)\\ Deal((A\|B)\nearrow C)=Deal(A)\cup Deal(B)\cup Deal(C)\\ Port((A\|B)\nearrow C)=\overline{Port(A)\cap Port(B)\cap Port(C)}\\ Extn(Port_{(A\|B)\nearrow C})=Extn(P_A)\cup Extn(P_B)\\ Publ(Port_{(A\|B)\nearrow C})=Publ(P_C)\\ Msg(s_{(A\|B)\nearrow C})=Msg(s_C)\\ Msg(r_{(A\|B)\nearrow C})=Msg(r_A)\cup Msg(r_B)\\ Cons(Port_{(A\|B)\nearrow C})=Cons(P_A)\cup Cons(P_B)\cup Cons(P_C) \end{cases} \tag{6}$$

定义 10. 设构件 A,B,C 是论域 $Dom(U)$中的 3 个构件,若$\exists s_A\in Publ(P_A)\wedge\exists s_B\in Publ(P_B)\wedge\exists r_C\in Extn(P_C)$,使得$[(per\text{-}cond(P_A)\wedge per\text{-}cond(P_C))\wedge(per\text{-}cond(P_B)\wedge per\text{-}cond(P_C))\wedge(s\Rightarrow r)]\wedge[Msg(s_A,s_B)\rightarrow Msg(r_C)]$,即构件 A、构件 B 通过发送消息“互斥”地“激发”构件 C 来实现功能需求,则称 A,B 与 C 进行了一次互斥“激发”运算,记作$(A\|B)\swarrow\!\!\nearrow C$.

如果把$(A\|B)\swarrow\!\!\nearrow C$ 看成一个整体,则$(A\|B)\swarrow\!\!\nearrow C$ 仍是一个构件,并且满足下列性质:

$$\begin{cases} Dom((A\,\|\,B)\swarrow^{\nearrow} C)=Dom(A)\cup Dom(B)\cup Dom(C) \\ Deal((A\,\|\,B)\swarrow^{\nearrow} C)=Deal(A)\cup Deal(B)\cup Deal(C) \\ Port((A\,\|\,B)\swarrow^{\nearrow} C)=\overline{Port(A)\cap Port(B)\cap Port(C)} \\ Extn(Port_{(A\|B)\swarrow^{\nearrow}C})=Extn(P_A)\cup Extn(P_B) \\ Publ(Port_{(A\|B)\swarrow^{\nearrow}C})=Publ(P_C) \\ Msg(s_{(A\|B)\swarrow^{\nearrow}C})=Msg(s_C) \\ Msg(r_{(A\|B)\swarrow^{\nearrow}C})=Msg(r_A)\cup Msg(r_B) \\ Cons(Port_{(A\|B)\swarrow^{\nearrow}C})=Cons(P_A)\cup Cons(P_B)\cup Cons(P_C) \end{cases} \tag{7}$$

定理 1. 异步“激发”运算可重构为两个并行的激发运算,即

$$C\updownarrow(A\|B)=(C\oplus A)\|(C\oplus B) \tag{8}$$

证明:要证明公式(8)成立,只需证明等式的左右两边满足定义 2 的 9 个条件即可.为此,我们证明具有代表性的条件 5、条件 6 成立,其他证明同理.

$Extn(Port(C\updownarrow(A\|B)))=Extn(P_C)$,把$(C\oplus A)\|(C\oplus B)$整体当成一个构件则会发现,它只有构件 C 的端口从外界环境获得消息,即 $Extn((C\oplus A)\|(C\oplus B))=Extn(P_C)$,左边等于右边,条件 5 成立.

$Publ(Port(C\updownarrow(A\|B)))=Publ(P_A)\cup Publ(P_B)$,把$(C\oplus A)\|(C\oplus B)$整体当成一个构件则会发现,它只有构件 A 和构件 B 的端口提供给外界环境消息,即 $Publ((C\oplus A)\|(C\oplus B))=Publ(P_A)\cup Publ(P_B)$,左边等于右边,条件 6 成立.其他条件均可同理得证,证毕. □

定理 2. 激发运算不满足交换律,即

$$A\oplus B\neq B\oplus A \tag{9}$$

证明:将 $A\oplus B$ 整体看作是一个构件,根据定义 4,记为 $A\Delta B$.其中, $Extn(P_{A\Delta B})=Extn(P_A), Publ(P_{A\Delta B})=Publ(P_B)$.将 $B\oplus A$ 整体看作是一个构件,根据定义 4,记为 $B\Delta A$.其中, $Extn(P_{B\Delta A})=Extn(P_B), Publ(P_{B\Delta A})=Publ(P_A)$.可见,左右两边并不相同,所以激发运算不满足交换律.证毕. □

定理 3. 两个并行的激发运算可重构为非互斥“激发”运算,即

$$(A\oplus C)\|(B\oplus C)=(A\|B)\swarrow^{\nearrow} C \tag{10}$$

证明:公式(10)的证明过程与公式(8)的证明过程相同,这里从略. □

1.3 可重构系统代数模型

下面给出可重构系统的定义.

定义 11. 设论域为 U:

(1) 构件是可重构系统的组成单元;

(2) 构件运算后的结果是构件;

(3) 构件经过有限次运算后组成可重构系统.

可重构系统记为 $S=\langle C,O\rangle$,简称重构系统.其中,C 表示构件集合,O 表示构件运算的集合,运算用 op_i 表示.

由定义 11 可知,重构系统的性质如下:

(1) 封闭性,即构件与构件、构件与重构系统、重构系统与重构系统经过运算后仍是重构系统;

(2) 层次性,即构件可由多个构件运算而成,多个构件经过运算可形成重构系统;

(3) 可扩展性,即一个满足条件的新构件可以通过运算加入到重构系统中.

从组合是构件间的运算实现角度可以进一步证明可重构系统对任意一个运算构成代数系统.

定理 4. 设 $S=\langle C,O\rangle$是可重构系统,则 S 对 O 中的每一个运算都构成代数系统.

证明:由可重构系统运算的封闭性,可得定理 4 的正确性. □

我们把 $S=\langle C,O\rangle$称为可重构系统的代数模型,也即可重构系统的代数表达式.

定义 12. 设 $S_1=\langle C_1,O_1\rangle, S_2=\langle C_2,O_2\rangle$是两个不同的重构系统,若$\forall m\in C_1,\exists n\in C_2$,使得 n 与 m 对应,则称 S_1 与 S_2

之间存在着一个映射,记为 $f:S_1\to S_2$.若映射是满射,则称可重构系统间是满射;若映射是一一对应,则称可重构系统间为一一映射.

定义 13. 设 $S_1=\langle C_1,O_1\rangle,S_2=\langle C_2,O_2\rangle$是两个重构系统,$f$是 S_1到 S_2的一个映射,若对$\forall x\in C_1$和$\forall y\in C_2$有 $f(xop_iy)=f(x)op_jf(y)$,其中,$op_i\in O_1,op_j\in O_2$,则称 f 为 S_1到 S_2的同态,即 S_1与 S_2同态.若 f 是单射,则称 f 是 S_1到 S_2的单一同态;若 f 是一一映射,则称 f 是 S_1到 S_2的同构,也称 S_1与 S_2同构.

同态和同构可以进一步描述可重构系统间的重构特性等概念.

定义 14. 给定两个重构系统 $S_1=\langle C_1,O_1\rangle,S_2=\langle C_2,O_2\rangle$,构造一个新的重构系统 $S_1\times S_2=\langle C_1\times C_2,op_k\rangle$.其中,$C_1\times C_2$是构件集合的笛卡尔积;$op_k$ 定义为:$\forall x_1,x_2\in C_1,\forall y_1,y_2\in C_2$,有$\langle x_1op_ix_2,y_1op_jy_2\rangle=\langle x_1,y_1\rangle op_k\langle x_2,y_2\rangle,op_i\in O_1,op_j\in O_2$.称 $S_1\times S_2$是 S_1到 S_2的重构结构,S_1和 S_2是 $S_1\times S_2$的重构因子,其中的 op_i,op_j,op_k是任意构件间的连接运算.

在定义 14 的基础上,可以定义重构系统的可重构对象.

定义 15. 对$\forall x_1,x_2\in C_1$和$\forall y_1,y_2\in C_2$,在定义 14 中,所有满足$\langle x_1op_ix_2\rangle$的$(x_1,x_2)$的任意一个子集称为 op_i的一个重构对象;所有满足$\langle y_1op_iy_2\rangle$的$(y_1,y_2)$的任意一个子集称为 op_j的一个重构对象.

2 可重构系统的可重构属性建模

可重构特性是可重构系统的重要属性,本节基于可重构系统的代数模型,讨论如何通过可重构范式对重构系统的可重构属性建模.

定义 16. 设 $S=\langle C,O\rangle$是一个重构系统,若$\exists Y\in C;\forall X\in C$ 都有 Xop_iY,则称 Y 是重构系统 S 的核.当核多于 1 个时,可选定其中一个作为主核.

根据构件的运算关系,我们可以判断一个重构系统是否存在核.因为重构过程中,构件间连接关系的变化可以导致重构系统由无核变为有核,或从有核变为无核.

定义 17. 设 $S=\langle C,O\rangle$是一个重构系统,若构件数目多于 1 个时,任意一个构件在执行时都至少与另外一个构件发生组合,则称 S 满足第一范式,记为 1NF.

显然,1NF 是可重构系统设计的最基本要求.

定义 18. 设 $S_1=\langle C_1,O_1\rangle$是一个重构系统,对 $x\in C_1$,若$\exists y\in Y,Y\subset C_1$,其中,$x\neq y$,使得 x 被 y 替换后,S_1仍为可重构系统,记作 S_1发生了构件替换重构.当同时用多个构件替换一个构件时,记作 S_1发生了构件添加重构.特别地,当 $y=\varnothing$时,记作 S_1发生了构件删除重构,添加重构和删除重构是替换重构的特例.为了保证系统功能的完整性,可重构系统中作为核的构件可以进行替换重构,不能进行删除重构.

定义 19. 设 $S_1=\langle C_1,O_1\rangle,S_2=\langle C_2,O_2\rangle$是两个重构系统,$f$是 S_1到 S_2的一个同构,$\exists A\in C_1\cap C_2,\exists B\in C_1\cap C_2$,若$\exists x\in C_1$,$y\in C_1,z\in C_1$,使得 $z\updownarrow(A\|B)=(f(x)\oplus A)\|(f(y)\oplus B)$,记作 S_2是 S_1的↕型重构.特别地,若一个重构系统 S 能够发生↕型重构,则称 S 满足↕范式.

定义 20. 设 $S_1=\langle C_1,O_1\rangle,S_2=\langle C_2,O_2\rangle$是两个重构系统,$f$是 S_2到 S_1的一个同构,$\exists A\in C_1\cap C_2,\exists B\in C_1\cap C_2$,若$\exists x\in C_2$,$y\in C_2,z\in C_2$,使得$(A\oplus f(x))\|(B\oplus f(y))=(A\|B)\nearrow z$,记作 S_2是 S_1的↗型重构.若一个重构系统 S 能够发生↗型重构,则称 S 满足↗范式.

可重构系统中的核构件是系统中进行重构的主要构件(如下面的案例 1 所示).在对一个系统进行重构分析时,找到系统的核构件,对我们判断系统是否可以重构具有重要意义,不仅可以简化分析,还有助于设计重构的类型.

引理 1. 任何一个可重构系统可以只用“激发”、“调用”和“并行”运算来表示,“激发”运算包含了单向、双向、同步、互斥类型.

定理 5. 任意一个 $S=\langle C,O\rangle$的代数表达式都可以表示成下面的标准型:

$$S=k_0(C_1\oplus C_2)\|k_1(C_3\otimes C_4)\|k_2(C_5\updownarrow(C_6\|C_7))\ldots\|k_3((C_m\|C_n)\swarrow\!\nearrow C_l)\ldots\|k_m(C_i\oplus C_j) \tag{11}$$

其中,$C_i\in C,C_j\in C,C_i\neq C_j,i=1,2,3,\ldots,n,j=1,2,\ldots,n.k(\updownarrow)$和 $k(\swarrow\!\nearrow)$表示同步和互斥型激发运算:

$$k_m=\begin{cases}1, & \text{operation exist}\\ 0, & \text{others}\end{cases},m=0,1,2,...,n$$

证明:用归纳法证明.假设 $m=0$ 时,$k_0=1,k_m=0,m=1,2,\ldots,n$ 成立,此时 $S=C_1\oplus C_2$,可重构系统由两个构件组成,构件间是激发运算,符合标准型.特别地,若 $C_2=\varnothing,S=C_1\oplus C_2=C_1$,可重构系统由一个构件组成.初始情况下,若 C_1,C_2,C_3 是同步激发或互斥激发,$S=C_3\updownarrow(C_1\|C_2)$或 $S=(C_1\|C_2)\swarrow\!\nearrow C_3$,则可重构系统由三构件组成,也符合标准型.

假设 $m=n-1,k_m=1,m=0,1,\ldots,n-1$ 时公式(11)成立,证明 $m=n$ 时公式(11)也成立.

即,$S=k_0(C_1\oplus C_2)\|k_1(C_3\otimes C_4)\|\ldots\|k_{n-1}(C_i\oplus C_j)$成立,证明 $S=k_0(C_1\oplus C_2)\|\ldots\|k_{n-1}(C_{n-2}\oplus C_{n-1})\|k_n(C_i\oplus C_j)$成立.

(1) 当 C_i,C_j 间是单向或双向激发运算时,$S=k_0(C_1\oplus C_2)\|\ldots\|k_{n-1}(C_{n-2}\oplus C_{n-1})\|k_n(C_i\oplus C_j)$显然成立;

(2) 当 C_i,C_j 参与异步激发运算时,因为 $C\updownarrow(A\|B)=(C\oplus A)\|(C\oplus B)$,即 $k_nC\updownarrow(A\|B)=k_n(C\oplus A)\|k_n(C\oplus B)$,等号右边是标准型,所以 $S=k_0(C_1\oplus C_2)\|\ldots\|k_{n-1}(C_{n-2}\oplus C_{n-1})\|k_n(C_i\oplus C_j)$成立.即添加的构件间是异步激发运算时,可以重构为使用“激发”和“并行”运算的构件.

(3) 当 C_i,C_j 参与非互斥激发运算时,因为$(A\oplus C)\|(B\oplus C)=(A\|B)\nearrow\!C$,即 $k_n(A\oplus C)\|k_n(B\oplus C)=k_n(A\|B)\nearrow\!C$,等号的左边是标准型,所以非互斥激发也是标准型,$S=k_0(C_1\oplus C_2)\|\ldots\|k_{n-1}(C_{n-2}\oplus C_{n-1})\|k_n(C_i\oplus C_j)$成立.即当添加的构件间是非互斥激发运算时,可以重构为使用“激发”和“并行”运算的构件.

(4) 当 $m=n$,添加的是 $k(\updownarrow)$或 $k(\swarrow\!\nearrow)$(同步或互斥型激发运算)时,因为它们就是标准型中的运算,因此可以直接添加到标准型中.

证毕. □

3 应用研究

文章提出的可重构代数模型和可重构范式可以应用到以下领域:(1) 可重构系统的形式化描述;(2) 可重构系统的体系结构设计;(3) 可重构系统的可重构属性分析等.

案例分析 1:运用前文中提出的重构范式,对系统的重构属性进行分析.图 1(a)是一个路由交换系统中安全服务构件组合示例,假设构件间不存在同步和互斥类型的激发运算.为了简化运算,我们将图 1(a)化简为图 1(b).

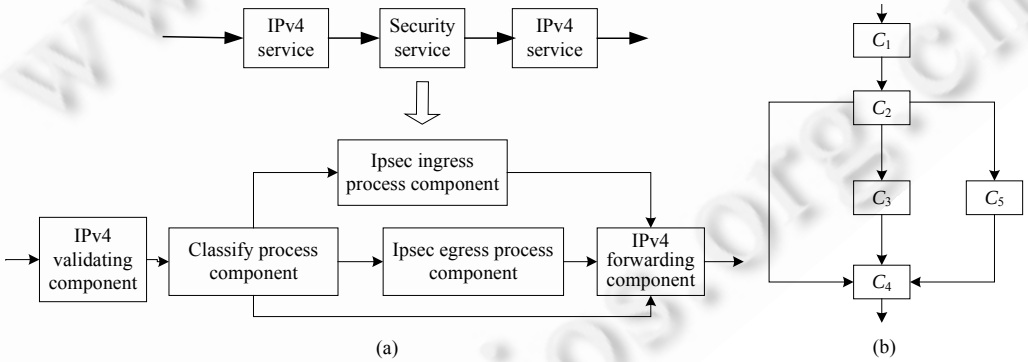


Fig.1 Security component combination for routers

图 1 路由交换系统中安全服务构件组合

根据前文对可重构代数模型的描述,图 1(b)可用表达式(12)表示:

$$C_1\oplus C_2\updownarrow(\varnothing\|C_3\|C_5)\nearrow\!C_4 \tag{12}$$

其中,$\varnothing$表示一条没有构件存在的消息路径;C_2 是系统的核构件,不难发现,公式(12)中,C_2 满足↗范式和↕范式.即,这个系统可以发生↗型重构和↕型重构.若应用↗范式可发生如下重构,如图 2 所示.

$$C_1\oplus C_2\updownarrow(\varnothing\|C_3\|C_5)\nearrow\!C_4=C_1\oplus(C_2\updownarrow((C_3\oplus C_4)\|(C_5\oplus C_4)\|C_4)) \tag{13}$$

若应用↕范式可发生如下重构,如图 3 所示.

$$C_1 \oplus C_2 \updownarrow (\varnothing \| C_3 \| C_5) \nearrow C_4 = C_1 \oplus ((C_2 \oplus C_3) \| (C_2 \oplus C_5) \| C_2) \nearrow C_4 \quad (14)$$

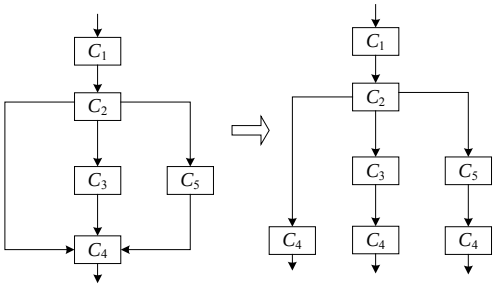


Fig.2 Schematic diagram for ↗ paradigm

图 2 重构系统↗范式示意图

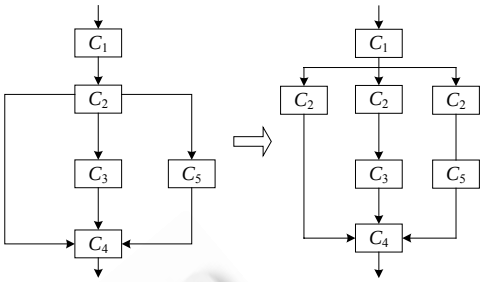


Fig.3 Schematic diagram for ↕ paradigm

图 3 重构系统↕范式示意图

案例 1 对一个路由交换系统中安全服务构件组合进行了重构属性分析,证明可以应用可重构代数模型发现系统可进行不同类别的重构.案例中:C_2 构件是系统的核;↗型重构后,C_2 还是系统的核;但进行↕型重构后,系统没有核.重构后的系统并不只是简单地增加几个构件,当对这些构件进行替换操作时,通过引入新类型的构件便可增加系统的功能.

案例分析 2:可重构系统的特点在于构件重构导致的系统功能发生变化,我们可以分析不同的系统间是否可以发生重构来设计可重构系统.

图 4(a)和图 4(b)表示以太网查表和 IP 路由查表使用的数据结构,我们使用存储构件来表示两者使用的存储系统.为了便于分析,我们将图 4(c)和图 4(d)转化为图 5 所示的构件组合示意图.其中,构件 C_1 和 C_2、C_3 和 C_4、C_5 和 C_6 之间均是调用运算.

图 5(a)的代数模型表达式如公式(15)所示:

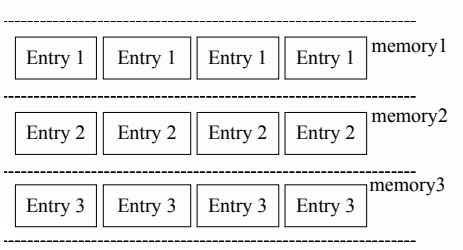
$$C_7 \updownarrow (C_1 \| C_3 \| C_5) \nearrow C_8 \quad (15)$$

图 5(b)的代数模型表达式如公式(16)所示:

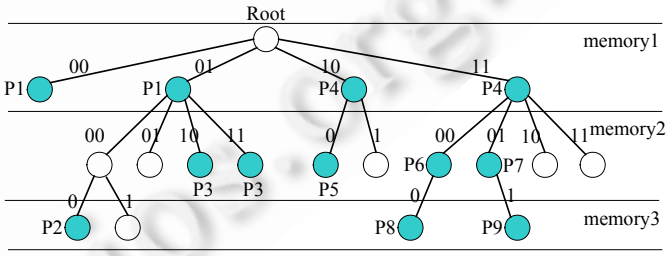
$$C_7 \oplus (C_1 \oplus C_3 \oplus C_5) \oplus C_8 \quad (16)$$

根据定理 5,可将公式(15)化为标准型,如公式(17)所示:

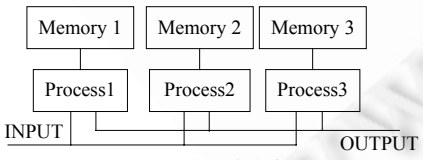
$$C_7 \oplus C_1 \oplus C_8 \| C_7 \oplus C_3 \oplus C_8 \| C_7 \oplus C_5 \oplus C_8 \quad (17)$$



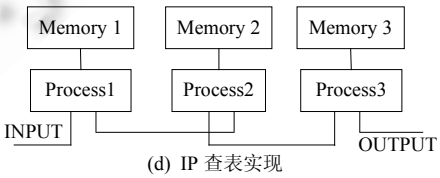
(a) 以太网哈希查表



(b) IP 多分支树查表



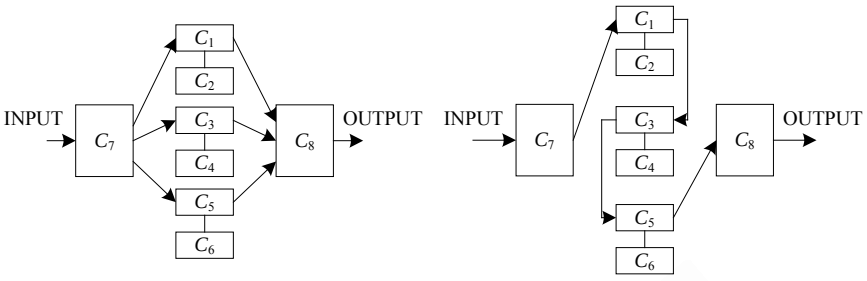
(c) 以太网查表实现



(d) IP 查表实现

Fig.4 Ethernet lookup and IP lookup

图 4 以太网查表系统和 IP 路由查表系统



(a) 以太网查表构件组合 (b) IP查表的构件组合

Fig.5 Component combination for Ethernet lookup and IP lookup

图 5 以太网查表和 IP 路由查表的构件组合示意图

从公式(16)和公式(17)中可以发现,这两个系统使用了相同的构件且不存在核,仅仅是构件间的运算关系不同.即,只要改动构件间的连接拓扑,就可以将一个支持以太网查表的系统重构为一个支持 IP 路由查表的系统.案例 2 以以太网查表和 IP 路由查表为例,演示了如何对两个系统进行可重构属性分析.通过将两个系统形式化为标准型,发现两者存在的重构属性,判断两者是否可以通过重构构件间的连接拓扑(构件运算)相互转化,从而可以在进行多功能查表系统的体系结构设计时考虑两种功能间的系统的重构属性设计,设计出具有重构能力的系统.

案例分析 3:构件的替换和删除是可重构系统最为常用的重构方式.

传统的系统程序设计时通过多分支的程序结构来满足多种处理模式的需要,这种设计方式要求系统设计阶段要考虑所有可能发生的情况.若在系统设计之初将分支处理的功能用构件的方式实现,当需求发生变化时,则可通过对构件的替换或删除来满足新的需求.这样做既满足需求的变化,又使系统具有了重构能力,可以适应更多的应用场景.

图 6 所示的是路由器中位于接口板卡中的业务识别模块,传统的路由器中使用不同的板卡对应不同的用户业务,例如电信板卡、广电板卡和互联网板卡等.当用户业务发生变化时,需要对路由设备更换板卡来适应新的业务.这样做带来很多不足,人为造成业务的中断,增加了业务的时延.若使用可重构系统,以图 6 中的业务识别模块为例,可以通过替换业务识别构件的方式来改变路由器对电信、广电和互联网等业务的支持.此时只发生了构件替换,不需要更换整个板卡.案例 3 仅仅显示了业务识别部分的替换重构,若想完整地实现电信业务到广电业务的切换,还需要替代其他与电信业务有关的构件.若未来出现新的业务,则可以用针对新业务的构件来替换现有构件.即用 C_n+C_{n+1} 替换现有的 C_n.这样,在保留原来业务的同时添加了对新业务的支持.

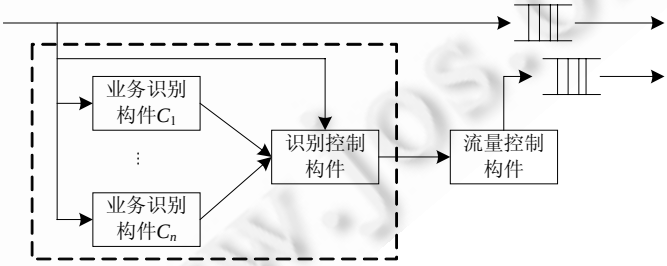


Fig.6 Identity module of routers

图 6 路由器中业务识别模块

4 相关研究工作

目前,对面向服务建模方法的研究还是集中在软件体系结构描述语言(architecture description language,简

称 ADL)方面.较著名的 ADL 有 ACME,C2,Darwin,Rapide 和 Wright 等[1].UML(union model language)是广泛使用的软件系统建模语言,虽然对 SA 的描述能力不足,但经过改进后仍然可以作为一种 ADL[2].Taylor[3]提出了一种可信 SA 描述语言 xADL,通过明确主题、资源、特权、安保和策略等相关概念,描述访问控制等安全事务.ADLs 采用形式化的方法描述 SA,但并未对组件、连接器和 SA 的属性和行为给出一致的定义和描述.Medvidovic[1]提出一个具有普遍意义的 SA 框架,但未对框架中的组件、连接器以及 SA 的属性和动态行为给出抽象化的描述,未对不同的连接方式进行分类,未讨论各类 SA 间的相互关系.从系统分析角度来看,SA 建模方法还应提供对质量属性的描述能力,而上述 ADLs 不具备这些功能.

还有研究人员对构件之间的交互行为展开研究,Allen[4],Magee 等人[5]运用 CSP、π演算等方法形式化地描述构件的行为交互协议,并显式地定义了系统的运行架构,但通常强调构件交互行为和系统架构的分析,缺少构件替换性的分析和验证算法.Canal 等人[6]运用π演算、Giese 等人[7]运用 Petri 网、Finkbeiner 等人[8]应用消息顺序图等描述构件的行为协议,侧重于行为协议的语法规约和兼容性的检测与分析.He 等人[9]对构件间组合关系进行研究,提出了构件演算的概念.他们将一个系统看作是多个互联的构件,建立了构件间的交互行为模型,分析了构件的可替代性.上述工作涉及了构件间连接拓扑可变和构件替换,但是都将构件约束为完整功能的实体,虽然构件可以进行替换,但是没有从整个系统功能可重构的角度考虑构件的划分和相互连接.

Beek[10],Brim 等人[11]提出用 team 自动机、交互自动机等描述构件的行为协议,注重系统的组装与交互特性,但未进行行为替换性分析.Attie 等人[12]提出了动态 I/O 自动机(dynamic I/O automata)模型描述和分析动态系统的行为,进行了构件替换性分析,但要求替换构件的迹包含于被替换构件.Tracz 等人[13,14]提出 3C 模型描述构件的交互行为,3C 模型从概念(concept)、内容(content)和语境(context)这 3 个方面对构件进行描述.概念,对构件功能进行抽象,说明构件做什么;内容,对抽象描述的实现,说明构件怎样做;语境,描述构件操作的环境,同时描述了构件复用所在的特定环境中必须满足的约束.

王千祥,杨芙清,吕建等人[15−17]提出了构件的自适应模型和理论,自适应模型的驻留性是指软件构件具有感知、收集外部环境、使用环境信息并对系统演化提供依据的能力;自适应模型的自主性是指软件构件在需要改变自身或整体行为或属性时能够相对独立、主动地开展变化,而无需按照面向对象方法那样所有行为都需要其他对象来驱动,缺乏独立性.自适应软件系统在外部环境和用户需求发生变化时,能够主动地改变自身的配置,使其自身更具有可重构性.

上述工作主要是如何准确、简洁地对构件间的连接关系进行形式化描述,从数学的角度描述一个多构件复杂系统的组成和运行特点,研究构件自适应模型和理论.本文的工作是通过将构件组合定义为构件的“运算”关系,研究系统的重构能力并定义重构范式,分析不同系统间重构的可能性和设计可重构系统.

5 结束语

本文用代数方法对面向服务的可重构系统进行了讨论,给出可重构构件、构件组合和可重构系统的形式化定义,提出了可重构系统的代数模型.在该模型的基础上给出了可重构范式概念,并提出多种重构范式,为可重构系统设计开发提供理论支持.上述研究中一个重要的观点是把构件组合解释成构件运算的实现,从而可以使用进程代数进一步讨论可重构系统的重构问题.可重构系统代数模型为进一步研究可重构理论问题奠定了基础.然而,可重构系统是一个复杂的系统,仅凭一个模型无法描述所有的特征,我们将继续研究其他建模方法.此外,可重构系统的可信性也是重要的研究内容之一,我们将在以后的研究中进一步展现可重构理论的应用价值.

References:

[1] Medvidovic N, Taylor RN. A classification and comparison framework for software architecture description languages. IEEE Trans. on Software Engineering, 2000,26(1):156−168. [doi: 10.1109/32.825767]

[2] Oquendo F. Formally modelling software architectures with the UML 2.0 profile for π-ADL. ACM SIGSOFT Software Engineering Notes Homepage Archive, 2006,31(1):1−13. [doi: 10.1145/1108768.1108773]

[3] Ren J, Taylor RN. A secure software architecture description language. In: Proc. of the Workshop on Software Security Assurance Tools, Techniques, and Metrics. Gaithersburg: National Institute of Standards and Technology, 2006. 82−90.

[4] Allen RJ. A formal approach to software architecture [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1997.

[5] Magee J, Kramer J. Dynamic structure in software architectures. In: Kaise GE, ed. Proc. of the 4th Symp. on the Foundations of Software Engineering (ACM SIGSOFT'96). New York: ACM Press, 1996. 3−14. [doi: 10.1145/239098.239104]

[6] Canal C, Fuentes L, Pimentel E, Troya JM, Vallecillo A. Extending CORBA interfaces with protocols. The Computer Journal, 2001, 44(5):448−462. [doi: 10.1093/comjnl/44.5.448]

[7] Giese H. Contract-Based component system design. In: Proc. of the 33rd Hawaii Int'l Conf. on Systems Sciences. Hawaii: IEEE Press, 2000. 125−134. [doi: 10.1109/HICSS.2000.927013]

[8] Finkbeiner B, Krüger I. Using message sequence charts for component-based formal verification. In: Proc. of the OOPSLA Workshop on Specification and Verification of Component-Based Systems. New York: ACM Press, 2001. 221−230.

[9] He JF, Liu ZM, Li XS. Component calculus. UNU/IIST Report, No.285, New York: UNU, 2003.

[10] Ter Beek MH, Ellis CA, Kleijn J, Rozenberg G. Synchronizations in team automata for groupware systems. The Journal of Collaborative Computing, 2003,12(1):21−69. [doi: 10.1023/A:1022407907596]

[11] Brim L, Černá I, Vařeková P, Zimmerova B. Component-Interaction automata as a verification-oriented component-based system specification. In: Proc. of the SAVCBS 2005. Lisbon, 2005. 31−38. [doi: 10.1145/1123058.1123063]

[12] Attie PC, Lynch NA. Dynamic input/output automata, a formal model for dynamic systems. In: Proc. of the 20th Annual ACM Symp. on Principles of Distributed Computing (PODC 2001). New York: ACM Press, 2001. 314−316. [doi: 10.1145/383962.384051]

[13] Tracz W. Implementation working group summary. Technical Report, Alexandria: Reuse in Practice Workshop Summary, 1990. 10−19.

[14] Gerosa M, Pimentel M, Fuks H. Development of groupware based on the 3C collaboration model and component technology. In: Proc. of the 12th Int'l Workshop (CRIWG 2006), Vol.4154. New York: Medina del Campo, 2006. 302−309. [doi: 10.1007/11853862_24]

[15] Wang QX, Shen JR, Mei H. An introduction to self-adaptive software. Computer Science, 2004,31(10):168−171 (in Chinese with English abstract).

[16] Yang FQ. Thinking on the development of software engineering technology. Journal of Software, 2005,16(1):1−7 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1.htm

[17] Lü J, Ma XX, Tao XP, Cao C, Huang Y, Yu P. Oriented Internetware model and the research of supporting technology. Science in China (E), 2008,38(6):1−37 (in Chinese).

附中文参考文献:

[15] 王千祥,申峻嵘,梅宏.自适应软件初探.计算机科学,2004,31(10):168−171.

[16] 杨芙清.软件工程技术发展思索.软件学报,2005,16(1):1−7. http://www.jos.org.cn/1000-9825/16/1.htm

[17] 吕建,马晓星,陶先平,曹春,黄宇,余萍.面向网构软件的环境驱动模型与支撑技术研究.中国科学(E 辑),2008,38(6):1−37.

袁博(1981－),男,山东济南人,博士生,主要研究领域为可重构网络,构件可重构技术.

汪斌强(1963－),男,博士,教授,博士生导师,主要研究领域为可重构网络.