基于逻辑推理的构件行为片段提取与重组研究

徐 俊[1] 肖 刚[1] 张元鸣[1] 高 飞[1] 方赵林[2]

(浙江工业大学计算机科学与技术学院 杭州 310023)[1] (浙江工业大学信息化办公室 杭州 310014)[2]

摘 要 从构件组装研究背景出发,针对软件应对复杂多变的业务需求时的不足,提出了一种基于逻辑推理的构件行为片段提取与重组算法。其主要思想是在研究构件接口映射、状态变迁的基础上,建立构件行为的结构模型和状态模型,并将其分解为基于π关系推导的构件行为片段。最后根据逻辑推理的方法,再将目标输入输出作为待求解,从关系推导中挖掘有效的行为片段进行重组,组装成一个满足目标需求的复合构件。

关键词 构件行为片段,π演算,逻辑推理,构件重组

中图法分类号 TP311 **文献标识码** A

Research on Component Behavior Fragment Extraction and Composition Based on Logical Reasoning

XU Jun[1] XIAO Gang[1] ZHANG Yuan-ming[1] GAO Fei[1] FANG Zhao-ling[2]

(College of Computer Science and Technology, School of Zhejian University of Technology, Hangzhou 310023, China)[1]

(Campus Information Center, Zhejiang University of Techology, Hangzhou 310014, China)[2]

Abstract In the background of components assembly, since the existing software can hardly fulfill changeable requirements of users, the paper proposed an algorithm for component behavior fragment extraction and composition based on logical reasoning. The main idea is based on the researches of interface mapping and state transition of components, which establishes the structural model and state model of component behavior, and breaks down then into component behavior fragments that are derived on the basic of π-calculus. In the end, following the method of logical reasoning, the paper took the input and output as target solutions, and provided a composite component to meet requirement goals by compositing the useful component behavior fragment from relation derivation.

Keywords Component behavior fragment, π-calculus, Logical reasoning, Component assembly

1 引言

随着软件技术的快速发展,构件技术成为大型复杂企业应用软件快速开发以及软件重构与复用的关键技术。同时,随着近年面向服务软件技术的研究和应用,如何提高软件的自组织性、适应性,以符合不同的功能、环境以及性能要求,从而满足用户目标需求,已成为研究的新领域。其中构件组装就是研究重点之一。

构件组装研究包括了对体系结构 SA(Software Architecture)的研究和对过程构件的形成研究等,如构件组装机制[1]、基于语义和知识的组装方法[2],或是对体系结构建模和高层描述的研究[3],国内取得一定进展的青鸟构件库 ABC 方法[4]研究就属于后者。目前的研究中多将组装中的构件作为原子结构对待,这是因为构件具有黑盒的特性,使用者通常无法了解构件的源代码和详细设计信息。同时,构件作为单一的功能实体,通过固定的接口提供服务,往往无法对其进行分拆。为了提高构件的检索效率和使用可靠性,人们开始利用构件的描述信息进一步剖析构件的内部逻辑。特别是构件功能行为的描述,因为它直观反映了构件能否满足用户的功能需求。目前,对于构件行为的形式化描述和分析验证是研究的热点。如文献[5]给出了一系列算法来检验行为的存在一致性和强制一致性,从而确定构件中是否具有用户想要的行为。文献[6]则通过对构件行为的形式化分析验证来测试构件功能的正确性。文献[7,8]则通过对行为的一致性检查来保证构件演化结果的正确性。但目前研究都集中于利用行为描述来验证已知构件组合的兼容性和正确性,并没有涉及对大量构件行为片段的挖掘和利用。在此背景下,本文提出了一种基于逻辑推理的构件行为片段提取与重组算法,通过对构件行为的分解,提取出有效的行为片段进行重组,以满足目标功能需求。

2 构件行为及其模型表示

构件体及其所在的环境包括其它构件之间所发生的信息交互以及构件体本身状态的变化,都被描述为一种构件行为。本文采用接口自动机对构件的行为进行建模。构件接口自动机的形式化定义如下:

到稿日期:2011-06-18 返修日期:2011-09-15 本文受国家自然科学基金项目(50705087),浙江省科技厅项目(2010C31002)资助。

徐 俊(1979-),男,硕士,实验师,主要研究方向为软件工程、电子商务,E-mail:xujun@zjut.edu.cn;肖 刚(1965-),男,教授,主要研究方向为 CAD/CAM、大规模定制设计理论、信息集成管理等;张元鸣(1977-),男,博士,讲师,主要研究方向为电子商务、软构件;高 飞(1974-),男,博士,副教授,主要研究方向为产品族设计、大批量定制;方赵林(1972-),男,硕士,副教授,主要研究方向为网络技术及应用。

定义 1 构件 C 的行为结构模型可以定义为 $ActionModel=(T,A,f_a)$，其中

(1)T 为该构件的端口集合。

(2)A 为构件动作集合，每个动作 a 是一个二元组 (p,m)。其中 p 为动作类型，为 *Send*(发送)或 *Receive*(接收)；m 为该动作携带的消息。

(3)f_a 为端口与动作的映射，指出了端口和动作之间的关联关系。

定义 2 构件 C 的构件行为状态模型可以定义为一个四元组 $StateModel=(S,s_0,S^f,R)$，其中

(1)S 为构件状态集合；

(2)s_0 为构件初始状态；

(3)S^f 为构件结束状态集合，一个构件只具有一个初始状态，但可以有多个结束状态，初始状态和结束状态可以为同一个状态；

(4)R 为状态的迁徙集合，其中每个状态迁徙 r 可以定义为一个五元组 $r=(s_s,s_e,c,a\langle m\rangle)$。其中 s_s 为迁移的起始状态；s_e 为迁移的目标状态；c 为迁移发生的要约条件；a 为触发迁移的动作；m 为动作携带的消息。

这里以一个工资账户管理构件 AccountManager 为例，给出构件的构件结构模型和行为状态模型。该构件供给管理员使用，构件拥有两个功能：1)根据指定的用户名修改用户工资账户；2)修改管理员的密码。如图 1 所示，构件 AccountManager 共 7 个接口，该构件在接收到管理员登录 *login* 的消息后，开始执行，其行为逻辑流程为：

(1)构件在初始状态 A 获取到 $Port_1$ 传入的登录 *login* 消息后进入状态 B。状态 B 对 *login* 信息进行验证。如果验证失败(即 false)，则直接迁移到结束状态 G，如果验证通过(即 true)，则等待用户进一步操作。

(2)当在 B 状态时，如果接口 $port_2$ 传入 *name* 消息，状态迁移到 C 并向 $port_3$ 返回 *account* 消息(指定用户名字的工资账户信息)，并迁徙到 D 状态(等待修改)。当 $port_4$ 传入 *edit* 消息后，状态迁移到 E(完成工资账户的修改)，然后再自动向 $port_5$ 返回 *success* 消息，并迁移到结束状态 G。

(3)当在 B 状态时，如果接口 $port_6$ 传入 *editpas* 消息，则状态迁移到 F(完成管理员密码修改)，并自动向 $port_7$ 返回 *success* 消息，并迁移到结束状态 G。

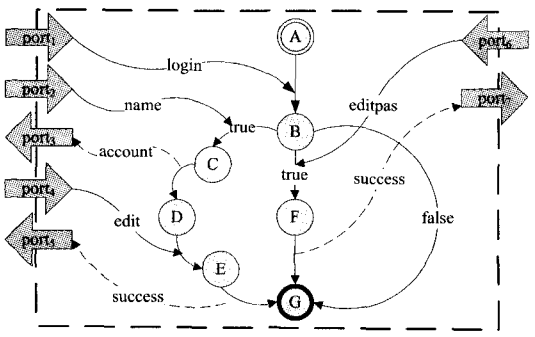

图 1 AccountManager 构件的行为视图

构件 AccountManager 的构件结构模型为 $ActionModel_{account}=(T,A,f_a)$，其中

$T=\{port_1,port_2,port_3,port_4,port_5,port_6,prot_7\}$

$A=\{a_1:(Receive,login),a_2:(Receive,name),a_3:(Send,account),a_4:(Receive,edit),a_5:(Send,success),a_6:(Receive,editpas),a_7:(Send,success)\}$

$f_a=\{(port_1,a_1),(port_2,a_2),(port_3,a_3),(port_4,a_4),(port_5,a_5),(port_6,a_6),(port_7,a_7)\}$

构件行为状态模型 $StateModel_{account}=(S,s_0,S^f,R)$，其中

$S=\{A,B,C,D,E,F,G\}$，

$s_0=A$，

$S^f=\{G\}$，

$R=\{r_1,r_2,r_3,r_4,r_5,r_6,r_7,r_8\}$，

$r_1=(A,B,\phi,Receive\langle login\rangle)$，

$r_2=(B,C,(login=true),Receive\langle name\rangle)$，

$r_3=(C,D,\phi,Send\langle account\rangle)$，

$r_4=(D,E,\phi,Receive\langle edit\rangle)$，

$r_5=(E,G,\phi,Send\langle success\rangle)$，

$r_6=(B,F,(login=\text{true}),Receive\langle editpas\rangle)$，

$r_7=(B,F,\phi,Send\langle success\rangle)$，

$r_8=(B,G,(login=\text{false}),\phi)$。

其中，触发动作为 ϕ，表示只要满足要约条件，迁移就自动发生。

3 构件行为片段抽取与重组

3.1 构件高阶型 π 演算进程描述

π 演算是计算机领域最重要的并发计算模型，由 Milner 等人[9]在 20 世纪 90 年代提出。而 Sangiorgi 则在此基础上提出了高阶型 π 演算[10]，用于描述并发系统中的结构和行为变化。只要将系统行为模型化为进程，构件和连接件的交互点模型化为通道，就可以将构件的行为显性地表示为一个 π 演算进程[9-11]。

高阶多型 π 演算进程可定义为：

$P::=0\mid ax.P\mid\bar{a}y.P\mid P+Q\mid P\mid Q\mid vxP\mid[x=y]P\mid D[\tilde{K}]$

其意义解释如下：

0 表示不进行交互的非活动进程，结束状态即为 0；

$ax.P$ 表示通过接口 a 输入接收消息 x，再迁移到状态 P；

$\bar{a}y.P$ 表示通过接口 a 输出信息 y，再迁移到状态 P；

$P+Q$ 表示根据条件迁移到 P 或者 Q，两者为或选择；

$P\mid Q$ 表示并行操作，指同时迁移到状态 P 和 Q，两者同时存在；

$vx.P$ 表示状态 P 不能通过接口 x 进行通信；

$[x=y]P$ 表示条件迁移，如果满足 $x=y$，则迁移到状态 P，否则将不进行任何操作，成为非活动进程 0；

D 为抽象的 $(\tilde{x})P$，表示以 $\tilde{x}$ 为参数的状态进程，$D\langle\tilde{K}\rangle$ 则是对 D 的具体化。

根据 π 演算理论，描述 AccountManager 的进行状态，同时为了便于表达，将信息以〈〉符号标识，表达式如下：

$P_{AM}^{\pi}=port_1\langle login\rangle.([login=\text{true}]port_2\langle name\rangle\overline{port_3}\langle account\rangle.port_4\langle edit\rangle.\overline{port_5}\langle success\rangle.0+[login=\text{true}]port_6\langle editpas\rangle.\overline{port_7}\langle success\rangle.0+[login=\text{false}].0)$

图 2 是一个工资支付 Wage 构件，该构件供管理员使用，功能是根据指定的工资账户支付工资。该构件在接收到管理员登录 login 的消息后开始执行。如登录成功，则等待 $port_2$

接收 *account* 消息后迁移到状态 C，接收到 $port_3$ 输入的工资信息 *pay* 后迁移到 D，并自动发出 *paysuccess* 消息，进入结束状态 E；如登录不成功，则直接进入结束状态 E。

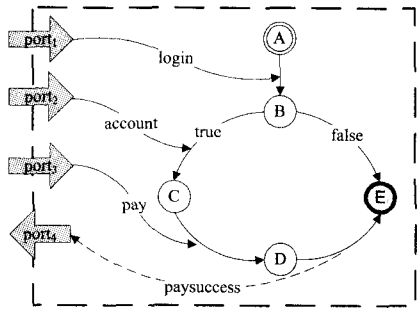

图 2　Wage 构件的行为视图

Wage 构件的 π 演算进程描述如下：

$$P^{\pi}_{Wage} = port_1\langle login\rangle.([login=\text{true}]\,port_2\langle account\rangle.\ port_3\langle pay\rangle.\ \overline{port_4}\langle paysuccess\rangle.\ 0 + [login=\text{false}].0)$$

如果服务目标是通过输入 login、name、pay 来完成工资支付 paysuccess，会发现通过单独的 AccountManager 和 Wage 构件均无法完成。同时，如果依据文献[12]的构件组合兼容性检查方法，AccountManager 和 Wage 因接口不匹配，也无法完成构件组合。因此本文提出了一种基于逻辑推理的构件行为片段提取与重组方法，通过提取构件中的有效行为片段，通过接口映射以及行为片段重组，获得有效输出，从而完成服务目标。

3.2　基于 π 关系推导的构件行为片段分解

首先根据进程的选择关系将 P^{π}_{Wage} 进行分解，扩展为 π 演算子进程描述，结果如下：

$$P^{\pi}_{Wage} = P^{\pi}_{Wage1} + P^{\pi}_{Wage2}$$

$$P^{\pi}_{Wage1} = port_1\langle login\rangle.\,[login=\text{true}]\,port_2\langle account\rangle.\ port_3\langle pay\rangle.\ \overline{port_4}\langle paysuccess\rangle$$

$$P^{\pi}_{Wage2} = port_1\langle login\rangle.\,[login=\text{flase}]0$$

$$P^{\pi}_{AM} = P^{\pi}_{AM1} + P^{\pi}_{AM2} + P^{\pi}_{AM3}$$

$$P^{\pi}_{AM1} = port_1\langle login\rangle.\,[login=\text{true}]\,port_2\langle name\rangle.\ \overline{port_3}\langle account\rangle.\ port_4\langle edit\rangle.\ \overline{port_5}\langle success\rangle$$

$$P^{\pi}_{AM2} = port_1\langle login\rangle.\,[login=\text{true}]\,port_6\langle editpas\rangle.\ \overline{port_7}\ success$$

$$P^{\pi}_{AM3} = port_1\langle login\rangle.\,[login=\text{false}]0$$

然后将构件的 π 演算子进程描述分解为行为片段 π 关系推导。

定义 3　构件行为片段 π 关系推导定义为 $P_{:}.m_a.[c]=P_{:}.m_o$，P 为所属进程，m_a 为输入，c 为传递条件，m_o 为输出，则 P^{π}_{Wage} 与 P^{π}_{AM} 推导式总集如下：

$$P^{\pi}_{Wage1}:.\ login.\,[login=\text{true}].\ account.\ pay = P^{\pi}_{Wage1}:.\ paysuccess \quad (1)$$

$$P^{\pi}_{Wage2}:.\ login.\,[login=\text{false}] = P^{\pi}_{Wage2}:.\ \phi \quad (2)$$

$$P^{\pi}_{AM1}:.\ login.\,[login=\text{true}].\ name = P^{\pi}_{AM1}:.\ account \quad (3)$$

$$P^{\pi}_{AM1}:.\ login.\,[login=\text{true}].\ name.\ edit = P^{\pi}_{AM1}:.\ success \quad (4)$$

$$P^{\pi}_{AM2}:.\ login.\,[login=\text{true}].\ editpas = P^{\pi}_{AM2}:.\ success \quad (5)$$

$$P^{\pi}_{AM3}:.\ login.\,[login=\text{false}] = P^{\pi}_{AM3}:.\ \phi \quad (6)$$

建立服务目标的 π 关系推导待求解为：

$$(P^{\pi}_{x}:.\ login.\,[login=\text{true}])(P^{\pi}_{y}:.\ name)(P^{\pi}_{z}:.\ pay) = P^{\pi}_{k}:.\ paysuccess(k\in\{x,y,z\}) \quad (7)$$

其中，x、y、z 为任一构件行为进程，由式(1)、式(3)可得：

$$\begin{aligned} P^{\pi}_{Wage1}:.\ paysuccess &= P^{\pi}_{Wage1}:.\ login.\,[login=\text{true}].\ account.\ pay \\ &= P^{\pi}_{Wage1}:.\ login.\,[login=\text{true}].\ (P^{\pi}_{AM1}:.\ login.\,[login=\text{true}].\ name).\ pay \\ &= (P^{\pi}_{Wage1}:.\ login.\,[login=\text{true}])(P^{\pi}_{AM1}:.\ login.\,[login=\text{true}])(P^{\pi}_{AM1}:.\ name)(P^{\pi}_{Wage1}:.\ pay) \end{aligned} \quad (8)$$

预设 *login* 同时满足 $P^{\pi}_{Wage1}:.\ login.\,[login=\text{true}]$ 与 $P^{\pi}_{AM1}:.\ login.\,[login=\text{true}]$，得

式(8)$=(P^{\pi}_{x}:.\ login.\,[login=\text{true}])(P^{\pi}_{y}:.\ name)(P^{\pi}_{z}:.\ pay)$

其中，$x=Wage_1\,|\,AM_1$，$y=AM_1$，$z=Wage_1$，x 为并行两个进程。由此得证，通过条件输入可以满足服务目标。

3.3　基于逻辑推理的组装算法

构件行为片段提取是在构件行为推导关系式上对行为输入输出进行匹配和替换，从而在候选构件中查找出能否达到服务目标的构件行为片段组合。前期研究[13]中给出了一种 ASABLR 算法，用于对构件的检索。在此对其进行扩展，将其应用到对构件行为的逻辑推理上，通过对输入输出的推理，提取出满足服务目标的行为片段并进行重组的方法。设候选构件为 $\{c_1, c_2, \cdots, c_n\}$，服务目标的可供输入为 $IN_{goal}=\{in_{goal1}, in_{goal1}, \cdots\}$，输出为 out_{goal}，具体推理过程描述如下：

(1)建立候选构件的 π 演算进程描述，然后将其转化为 π 关系推导式集 kc_{ai}，a 为构件序号，i 为进程序号，其中输入表示 in_{aij}，输出为 out_{aij}；

(2) 遍历 kc_{ai}，如不存在. $out_{aij}=$. out_{goal}，表示无解，结束推理；

(3) 遍历 kc_{ai}，对于某推导. $in_{aij}\cdots=$. out_{aij}，如其所有输入均满足 $in_{aij}\in IN_{goal}$，则表示通过可供输入可以得到该推导的输出. out_{aij}，将. out_{aij} 保存至 IN_{goal}，并从 kc_{ai} 中删除该推导，进入步骤(4)；如遍历后无任何推导满足条件，表示无解，结束推理；

(4) 检查 IN_{goal} 是否已经存在 out_{goal}，如已存在，表示求解成功，进入步骤(5)，否则返回步骤(4)；

(5) 根据 IN_{goal} 中的 out_{aij} 与 in_{goali}，反向建立输入输出映射，完成构件行为组合方案。

以式(7)为例进行逻辑推理，IN_{goal} 变化如下：

(1)$(P^{\pi}_{x}:.\ login.\,[login=\text{true}])(P^{\pi}_{y}:.\ name)(P^{\pi}_{z}:.\ pay)$；

(2)$(P^{\pi}_{x}:.\ login.\,[login=\text{true}])(P^{\pi}_{y}:.\ name)(P^{\pi}_{z}:.\ pay)(P^{\pi}_{AM1}:.\ account)$；

(3)$(P^{\pi}_{x}:.\ login.\,[login=\text{true}])(P^{\pi}_{y}:.\ name)(P^{\pi}_{z}:.\ pay)(P^{\pi}_{AM1}:.\ account)(P^{\pi}_{Wage1}:.\ paysuccess)$。

满足服务目标输出 $P^{\pi}_{k}:.\ paysuccess$，构件行为组合成功。

本算法将服务目标的输入作为已知条件，克服了计算的盲目性，提高了正向推理的效率，同时利用结果的反向推导，建立组合方案。该算法在行为片段为 n 的情况下，复杂度为 $O(n^2)$。

3.4　基于接口映射的构件行为片段重组

由 3.2 节可知，通过 AccountManager 和 Wage 构件的行为片段重组可以达到服务目标。但是构件体本身仍有服务目标不需要的行为片段，因此可以通过建立具有接口映射的复

合构件提供服务目标所需服务。根据 3.2 节中服务目标对 AccountManager 和 Wage 构件建立的复合构件 AccountWage 行为视图如图 3 所示。

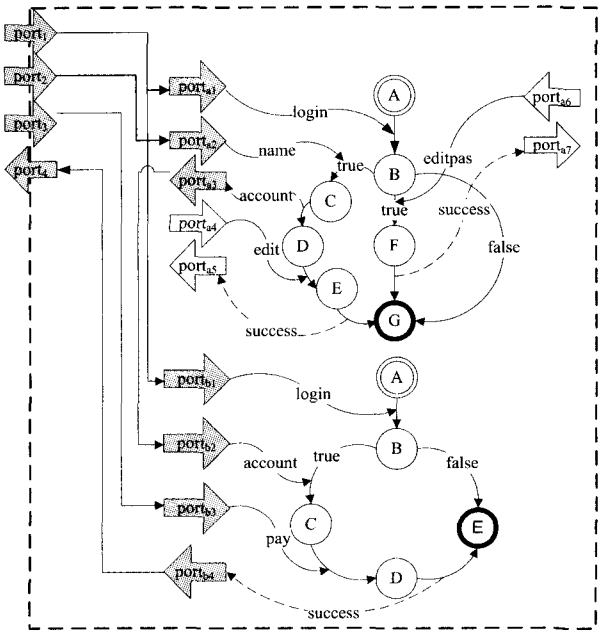

图 3 复合构件 AccountWage 行为视图

结束语 本文通过研究构件行为，介绍了行为的 π 演算进程描述，并以此建立行为片段的 π 关系推导，实现对行为的片段分解。提供了一种基于逻辑推理的构件行为片段提取与重组算法，根据服务目标所提供的输入、输出作为待求解，从候选构件中挖掘能满足目标需求的行为片段，通过行为片段重组，满足服务目标需求，最后通过实例证明了本方法的有效性。

参 考 文 献

[1] 任洪敏，钱乐秋. 构件组装及形式化推导研究[J]. 软件学报，2003，14(6)：1066-1074

[2] Sadaoui S. Composition of structured process specifications [J]. Electronic Notes in Theoretical Computer Science，2003，82(5)：132-143

[3] Allen R，Garlan D. A formal basis for architectural connection [J]. ACM Transactions on Software Engineering and Methodology，1997，6(3)：213-249

[4] 梅宏，陈锋，冯耀东，等. ABC：基于体系结构、面向构件的软件开发方法[J]. 软件学报，2003，14(4)：721-732

[5] 胡军，于笑丰，张岩，等. 基于场景规约的构件式系统设计分析与验证[J]. 计算机学报，2006，29(4)：513-525

[6] 李良明，王志坚，唐龙业. 构件功能行为测试的研究[J]. 小型微型计算机系统，2010，31(4)：686-690

[7] Hameurlain N. A Formal Framework for Component Protocols Behavioral Compatibility[C]// Proceedings of the 13th Asia Pacific Software Engineering Conference. Banglore：IEEE Computer Society，2006：87-94

[8] 罗毅，李兴宇，关连伟. 构件演化中的系统行为一致性的研究[J]. 计算机科学，2008，35(1)：266-270

[9] Milner R，Parrow J，Walker D. A calculus of mobile processes [J]. Information and Computation，1992，100(1)：1-40

[10] Sangiorgi D. Expressing mobility in process algebras：First-order and higher-order paradigms[D]. Edinburgh：University of Dinburgh，1992

[11] 李长云，李赣生，何频捷. 一种形式化的动态体系结构描述语言[J]. 软件学报，2006，15(6)：1349-1359

[12] 张弛. 基于行为描述的软件构件组合兼容性检查[J]. 计算机工程，2010，36(12)：46-51

[13] 肖刚，王培君，陆佳炜，等. 基于逻辑推理的构件组装策略及其算法[J]. 信息与控制，2009，38(6)：759-764

(上接第 109 页)

[4] Chuk Y. A Quantitative Methodology for Software Risk Control [C]// IEEE International Conference on Systems Management and Cybernetics. San Antonio，TX，1994：2015-2020

[5] [美]Boehm B W. 软件工程经济学[M]. 李师贤，等译. 北京：机械工业出版社，2004

[6] Stuart A，Massimo F. Quantitative Aspects of Requirements Evolution[C]// The 26th Annual International Computer Software and Application Conference. Oxford，England，2002：27-32

[7] Yan Yu-qing，Li Shi-xian，Liu Xian-ming. Quantitative Analysis for Requirements Evolution's Ripple-Effect [C]// Bangkok，Thailand：IEEE Computer Society. Los Alamitos，CA，US，2009：423-427

[8] Li Yin，Li Juan，Yang Ye，et al. Requirement-Centric Traceability for Change Impact Analysis：A Case Study [C]// Making Globally Distributed Software Development a Success Story. Berlin / Heidelberg：Springer，2008：100-111

[9] Yan Yu-qing，Li Shi-xian，Sun Wei-jun. Dependency based technique for identifying the ripple effect of requirements evolution [J]. Being accepted by Journal of Software

[10] Lavazza L，Valletto G. Requirements-based Estimations of Change Costs[J]. An International Journal of Empirical Software Engineering，2000，5(3)：229-243

[11] Fenton N E，Neil M. Software Metrics：Roadmap[C]// ICSE'00 Proceedings of the Conference on The Future of Software Engineering. 2000：357-370

[12] Loconsole A. Measuring the Requirements Management Key Process Area[C]// Proceedings of ESCOM-European Software Control and Metrics Conference. London，UK，April 2001

[13] Hammer T F，Huffman L L，Rosenberg L H. Doing requirements right the first time[J]. The Journal of Defence Software Engineering，1998：20-25

[14] Loconsole A. ICSE'04 Proceedings of the 26th International Conference on Software Engineering[C]// 2004：42-44

[15] 陆传赛. 排队论(第 2 版)[M]. 北京：北京邮电大学出版社，2009

[16] 陆凤山. 排队论及其应用[M]. 长沙：湖南科学技术出版社，1984

[17] [美]华兴. 排队论与随机服务系统[M]. 上海：上海翻译出版社，1987

[18] 唐应辉，唐小我. 排队论——基础与分析技术[M]. 北京：科学出版社，2006

[19] Dahlstedt Å G，Persson A. Requirements Interdependencies-Moulding the State of Research into a Research Agenda[C]// The Ninth International Workshop on Requirements Engineering：Foundation for Software Quality (REFSQ 2003)，held in conjunction with CAiSE，2003：71-80

[20] [美] Wiegers K E. 软件需求(第 2 版)[M]. 刘伟琴，刘洪涛，译. 北京：清华大学出版社，2004

[21] [美] Kovitz B L. 实用软件需求(第 1 版)[M]. 胡辉良，张罡，译. 北京：机械工业出版社，2005