软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(11):2668−2683 [doi: 10.3724/SP.J.1001.2011.03926]

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

基于有限自动机的多层次构件行为匹配模型*

初佃辉[1], 孟凡超[1+], 战德臣[1,2], 徐晓飞[1,2]

[1](哈尔滨工业大学(威海) 计算机科学与技术学院,山东 威海 264209)

[2](哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

Multi-Level Component Behavior Matching Model Based on Finite Automata

CHU Dian-Hui[1], MENG Fan-Chao[1+], ZHAN De-Chen[1,2], XU Xiao-Fei[1,2]

[1](School of Computer Science and Technology, Harbin Institute of Technology at Weihai, Weihai 264209, China)

[2](School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: E-mail: mengfanchao74@163.com

Chu DH, Meng FC, Zhan DC, Xu XF. Multi-Level component behavior matching model based on finite automata. *Journal of Software*, 2011,22(11):2668−2683. http://www.jos.org.cn/1000-9825/3926.htm

Abstract: The aim to improve deficiency of current research on components retrieval is based on behavior specification matching, a multi-level component behavior matching model based on finite automata, which is presented in this paper. The study uses finite automata to model the behavior of components, refers to the idea of graph matching in graph theory, proposes six kinds of behavior matching relationships: equivalence behavior matching, extended behavior matching, compatible behavior matching, contain behavior matching, weak contain behavior matching and weak compatible behavior matching, analysis the implication relationships among these behavior matching relationships, and gives corresponding decision algorithms and adaptation methods of each behavior matching relationship. Based on these algorithms, a universal decision algorithm is proposed to reduce the complexity of subsequent component adapter and assembly. The matching model proposed in this paper provides a favorable technical support for components retrieval, based on behavior specification matching.

Key words: finite state automata; component; behavior matching; behavior mapping graph

摘 要: 针对目前基于行为规约匹配的构件获取方面的不足,提出了一种基于有限自动机的多层次的构件行为匹配模型.该模型采用有限自动机对构件的行为进行建模,借鉴图论中图匹配的思想,提出了6种行为匹配关系:等价行为匹配、扩展行为匹配、相容行为匹配、包含行为匹配、弱包含行为匹配和弱相容行为匹配.分析了这些行为匹配关系之间的蕴涵关系,并给出各种行为匹配关系的判定算法和相应的适配方法.基于这些算法,提出了一种通用的行为匹配判定方法,以此来减少后继构件适配和组装的复杂性.所提出的匹配模型为基于行为的构件获取提供了有利的技术支持.

关键词: 有限自动机;构件;行为匹配;行为映射图

中图法分类号: TP311 **文献标识码**: A

* 基金项目: 国家自然科学基金(61033005); 国家高技术研究发展计划(863)(2008AA404Z101); 山东省科技攻关项目(2011GGX10108, 2010GGX10104, 2010GGX10116, 2010GZX20126)

收稿时间: 2009-12-25; 修改时间: 2010-05-12; 定稿时间: 2010-07-28

基于构件的软件开发(component-based software development,简称 CBSD)是解决大型复杂软件系统快速开发和软件重构与复用的关键技术.近年来,随着面向服务体系结构(service oriented architecture,简称 SOA)和 Web 服务技术的兴起,构件的接口被发布为 Web 服务,以支持大规模的基于 Internet 的构件开发[1],进一步促进了 CBSD 的发展与应用.CBSD 不同于传统的软件开发方法,它着重强调可复用构件的获取与组装,即用现成的构件来开发新的应用系统.构件的获取是基于构件的软件复用的一个重要内容,尽管目前已经提出了许多构件获取方法,然而这些方法大都是基于构件属性值的获取,而没有考虑构件的行为,因此很难指导后继构件适配与组装过程中行为的验证.本文采用有限自动机来描述构件的行为语义,并提出了一种基于有限自动机的多层次构件行为匹配模型,用以简化构件适配与组装过程中行为的验证复杂性,以此来提高构件的复用效率.

1 相关工作

目前,构件获取的方法可以分为 4 类:基于关键词的方法、枚举的方法、基于刻面的方法和基于规约匹配的方法.基于关键词的方法是利用 1 个或多个关键词来表示构件,其优点是表示方便、容易维护、便于检索,但是缺乏描述复杂构件的能力[2].枚举方法是采用一种自顶向下、逐层分解的方法将每一个应用领域分解成若干个相互关联的概念,每一软件制品是根据事先定义好的领域知识进行描述[3].它的优点是,其分层结构可以很容易地描述概念之间的关系.然而这种方法也存在许多问题,例如,缺乏柔性、不易维护等.刻面方法是把软件制品分解成一些基本术语,并将其组织成若干个刻面,每个刻面描述了软件制品的一个方面[4].该方法具有柔性高、扩展性好以及适应能力强等优点,但同时存在刻面使用效率不高的问题.另外,该方法对构件的行为特征的描述也不够精确.基于规约的方法采用形式化规格说明来描述构件的语义[5].构件包含很多语义信息,语义抽象既能精确地描述构件的行为,又能说明构件的功能.语义抽象的两个典型例子是基调描述和行为描述.基于规约的方法能够准确地刻画构件的特征,因此能够有效地指导构件的适配与组装.

本文主要研究基于行为规约匹配的构件获取.目前,描述构件行为的方法主要有:有限自动机[6−12]、Petri net[11]、进程代数[12]和π演算[13]和规则表达式[14].与其他 3 类方法相比,采用有限自动机描述构件的行为具有如下优点:

(1) 有限自动机更加适合描述面向对象范型的构件模型的行为,而目前大多构件都是采用面向对象语言实现.

(2) 有限自动机的复杂度较低,能够有效地检查行为的兼容性、可替代性和互操作性等性质.而采用 Petri net、进程代数和π演算,其计算复杂度非常高.

因此,本文采用有限自动机对构件的行为进行建模.该方法借鉴了接口自动机[6]的思想,并对其进行扩展,增加了终止状态.当前,基于有限自动机的构件获取方法也有不少研究成果.Rebeca 等人[7]在其 ARIFS 工具中使用 SCTL 描述构件规约,SCTL 采用状态转换图来描述构件的行为语义,并给出了相应的分类、检索、度量与适配方法.Andreas 等人[8]采用带有标注的有限状态自动机来建模业务流程构件,业务流程构件之间的匹配被定义为有限状态自动机之间的交集.Grigori 等人[9]将 Web 服务构件抽象为一个图结构,使用图的错误纠正匹配算法来计算 Web 服务之间的匹配度.Mahleko 等人[10]将有限自动机转换为一个可以进行索引的语法结构,以此来提高构件获取效率.由于行为描述的复杂性,因此,基于行为规约的匹配方法要求具有一定的松弛能力,以支持用户需求行为的不精确匹配.然而,现有的基于行为规约匹配方法大都只考虑构件之间的某种特定的行为匹配关系,例如,文献[7,8,10,11]主要考虑了相容行为匹配,而没有考虑更松弛的匹配关系.文献[7]虽然给出了行为匹配度,但是仅有匹配度还不能指导后继基于行为的构件适配和组装.

为了支持不精确的构件获取,本文在综合已有研究成果的基础上提出了 6 种行为匹配关系,由强到弱依次为:等价行为匹配、扩展行为匹配、包含行为匹配、相容行为匹配、弱包含行为匹配和弱相容行为匹配,它们构成了一个分层次的柔性构件行为匹配模型.针对每种行为匹配关系,我们给出了相应的判定算法和行为适配方法.本文所提出的方法为基于行为匹配的构件获取提供了有力的理论与技术支持.

2 基于有限自动机的构件行为模型

本文采用有限自动机对构件的行为进行建模.下面我们给出基于有限自动机构件行为模型的形式化定义.

定义 1(构件行为模型). 构件 C 的行为模型可以定义为五元组:$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$,其中,$S_C$ 为有限状态的集合; $\Sigma_C=\Sigma_C^I\cup\Sigma_C^H\cup\Sigma_C^\rho$ 为动作的集合,动作是指对构件所包含的操作的调用或者返回操作的执行结果, Σ_C^I 为输入动作的集合, Σ_C^H 为内部动作的集合, Σ_C^ρ 为输出动作的集合;$T_C=S_C\times\Sigma_C\times S_C$ 为状态迁移的集合;$s_{C0}\in S_C$ 为初始状态,每个构件只有 1 个初始状态,构件由此开始执行一个动作;$S_{CF}\subseteq S_C$ 为终止状态集,当构件达到终止状态时,执行过程结束,每个构件可以具有多个终止状态.

可以用 $s\xrightarrow{a}s'$ 表示 P_C 的一个迁移,其中,$s,s'\in S_C,a\in\Sigma_C,(s,a,s')\in T_C$.可以用 $FS(s)=\{s'|s'\xrightarrow{a}s\in T_C,s'\in S_C,a\in\Sigma_C\}$表示状态 s 的前驱状态集,$NS(s)=\{s'|s\xrightarrow{a}s'\in T_C,s'\in S_C\}$表示状态 s 的后继状态集,$FT(s)=\{s'\xrightarrow{a}s|s'\in S_C\}$表示状态 s 的前驱迁移集,$NT(s)=\{s\xrightarrow{a}s'|s'\in S_C\}$表示状态 s 的后继迁移集.如果对于$\forall s\xrightarrow{a}s'$, $s\xrightarrow{a}s''.s'=s''$,则称 P_C 为确定有限自动机;否则,称 P_C 为非确定有限自动机.本文约定 P_C 为确定有限自动机.

定义 2(执行片段和运行). P_C 的一个执行片段可以定义为一个状态与动作交替的有穷序列,记为

$$\eta=s_0\xrightarrow{a_0}s_1\xrightarrow{a_1}s_2\ldots s_{n-1}\xrightarrow{a_{n-1}}s_n,$$

其中,$s_i\xrightarrow{a_i}s_{i+1}\in T_C(0\leqslant i\leqslant n-1)$,$s_0$ 为 P_C 的初始状态.

如果 $s_n\in S_{CF}$ 为 P_C 的一个终止状态,则称η为 P_C 的一个运行.

定义 3(迹、字和接受语言). 设 $\eta=s_0\xrightarrow{a_0}s_1\xrightarrow{a_1}s_2\ldots s_{n-1}\xrightarrow{a_{n-1}}s_n$ 为 P_C 的一个执行片段,由η上所有动作组成的序列称为η的一个迹,记为 $trace(\eta)=a_0,a_1,\ldots,a_{n-1}$.如果$\eta$为一个运行,则称 $trace(\eta)=a_0,a_1,\ldots,a_{n-1}$ 为 P_C 所接受的合法动作序列,或者称为 P_C 的一个字.可以用 $L(P_C)$表示能够被 P_C 所接受的合法动作序列集,或者称为 P_C 所接受的语言.

图 1 描述了一个接收客户订单(*Receive_Order*)构件的行为模型.*Receive_Order* 的执行过程如下:首先接收客户所传递的订单(*rec_order*),然后进行库存检查(*chk_inv*),如果库存可用(*chk_inv*:*ok*),则确认订单(*confirm*);如果库存不可用(*chk_inv*:*fail*),则取消订单(*cancel*).这里,“?”表示输入动作,“;”表示内部动作,“!”表示输出动作. *Receive_Order* 包含 6 个状态,其中,状态 0 为初始状态,状态 5 为终止状态.根据执行片段、运行、迹和字的定义, $\eta_1=0\xrightarrow{rece\_order}1\xrightarrow{chk\_inv}2$ 为 *Receive_Order* 的一个执行片段,$trace(\eta_1)$=*rece_order*,*chk_inv* 为 *Receive_Order* 的迹. $\eta_2=0\xrightarrow{rece\_order}1\xrightarrow{chk\_inv}2\xrightarrow{chk\_inv:ok}3\xrightarrow{conform}5$ 为 *Receive_Orde* 的一个运行,$trace(\eta_2)$=*rece_order*, *chk_inv*,*chk_inv*,*chk_inv*:*ok*,*chk_inv*:*fail*,*confirm* 为 *Receive_Orde* 所接受的一个字.

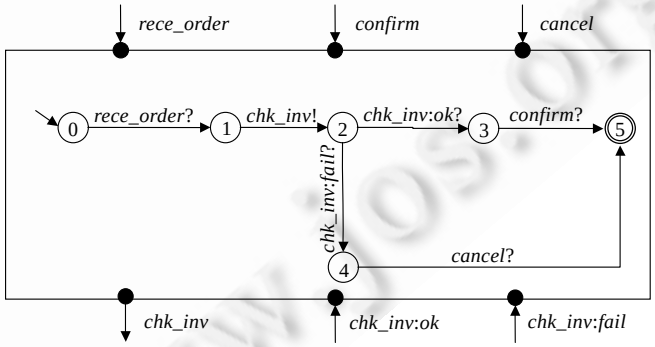

Fig.1 Receiving customer order component

图 1 接收客户订单构件

3 多层次构件行为匹配模型

行为匹配是比较用户需求的行为模型与构件的行为模型之间的匹配关系.本文提出了 6 种行为匹配关系:

等价行为匹配、扩展行为匹配、包含行为匹配、弱包含行为匹配、相容行为匹配和弱相容行为匹配.下面,我们给出每种行为匹配关系的定义.

定义 4(等价行为匹配). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,C 为构件库中的一个构件,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.如果存在一个从 S_Q 到 S_C 的同构映射 $f:S_Q\to S_C$,满足如下 3 个条件,则称 C 为 Q 的一个等价行为规约,记为 $C\xrightarrow{Equi\text{-}Beha}Q$:

(1) 对于 Q 中的任意一个迁移 $s_Q\xrightarrow{a}s'_Q\in T_Q$,在 C 中存在一个迁移 $f(s_Q)\xrightarrow{a'}f(s'_Q)$,满足条件 $a\sim a'$;并且对于 C 中的任意一个迁移 $s_C\xrightarrow{a'}s'_C\in T_C$,在 Q_P 中存在一个迁移 $f^{-1}(s_C)\xrightarrow{a}f^{-1}(s'_C)$,满足条件 $a\sim a'$.

(2) $f(s_{Q0})=s_{C0}$.

(3) 如果 $s_Q\in S_{QF}$,则 $f(s_Q)\in S_{CF}$.

在上述定义中,$a\sim a'$表示动作 a'与 a 的行为等价,即两个动作的签名具有相同的语义.由于本文主要研究构件行为协议级别的行为匹配,因此我们将动作看成一个具有语义标签的原子行为单元.同构映射要求两个行为模型具有结构完全相同的状态图,状态可能具有不同的名称,但是如果重新标识,两个状态图的结构完全一致.图 2 给出一个等价行为匹配的实例,在本例中,$f(0)=0,f(1)=1,f(2)=2,f(3)=4,f(4)=3,f(5)=5$.

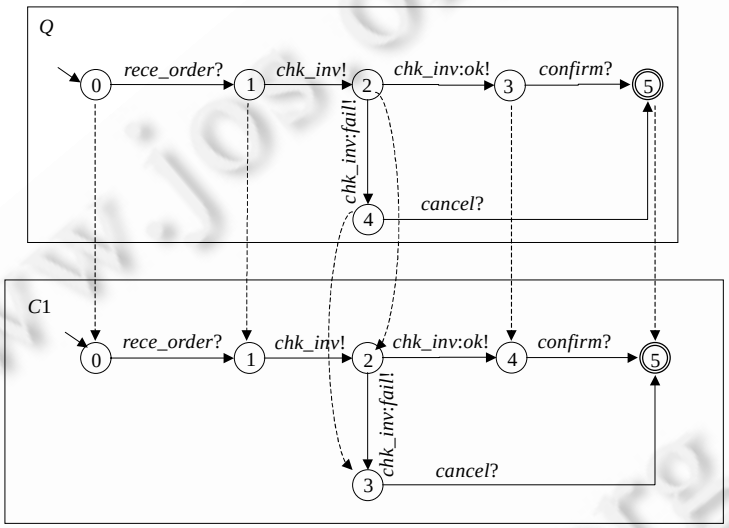

Fig.2 Equivalent behavior matching

图 2 等价行为匹配

定义 5(扩展行为匹配). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,C 为构件库中的一个构件,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.如果存在一个从 S_Q 到 S_C 的二元关系 $f\subseteq S_Q\times S_C$,满足如下 4 个条件,则称 C 为 Q 的一个扩展行为规约,记为 $C\xrightarrow{Exten\text{-}Beha}Q$:

(1) 对于任意 $s_Q\in S_Q$,存在 $s_C\in S_C$,满足条件$(s_Q,s_C)\in f$.

(2) 对于 Q 中的任意一个迁移 $s_Q\xrightarrow{a}s'_Q\in T_Q$,在 C 中存在一个迁移 $s_C\xrightarrow{a'}s'_C\in T_C$,满足条件:

$$(s_Q,s_C)\in f\wedge(s'_Q,s'_C)\in f\wedge(a\sim a').$$

(3) $(s_{Q0},s_{C0})\in f$.

(4) 对于任意 $s_Q\in S_{QF}$,存在 $s_C\in S_{CF}$,满足条件$(s_Q,s_C)\in f$.

根据扩展行为匹配关系的定义,如果 $C\xrightarrow{Exten\text{-}Beha}Q$,则说明 C 在行为上能够满足 Q 的需求,同时可能还存在冗余行为.图 3 给出了一个扩展行为匹配的实例,在本例中,$f=\{(0,0),(1,1),(2,4),(3,5),(4,6),(5,7)\}$.

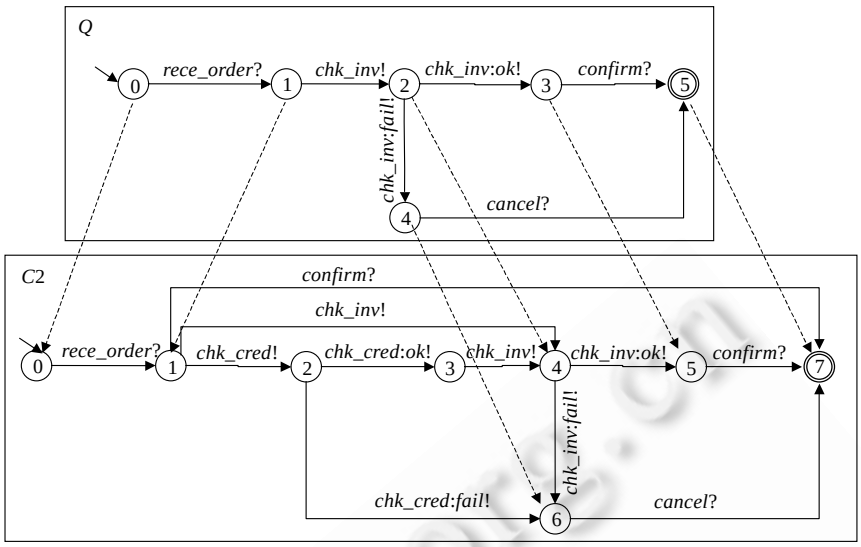

Fig.3 Extended behavior matching

图 3 扩展行为匹配

等价和扩展行为匹配的要求比较严格.下面,我们给出 4 种更加松弛的行为匹配.

定义 6(相容行为匹配). 设 Q 为一个需求构件,C 为构件库中的一个构件.如果存在字 $p=a_1,a_2,\ldots,a_n \in L(Q)$和 $p'=a'_1,a'_2,...,a'_n \in L(C)$,满足条件 $a_i \sim a'_i$ $(1 \leqslant i \leqslant n)$,则称 C 为 Q 的一个相容的行为规约,记为 $C \xrightarrow{Comp\text{-}Beha} Q$.

根据相容行为匹配关系的定义,如果 $C \xrightarrow{Comp\text{-}Beha} Q$, 则说明 C 中至少存在 1 个字在行为上能够满足 Q 的需求.图 4 给出了一个相容行为匹配的实例.在本例中,构件 $C3$ 中的字:*rece_order*,*chk_inv*,*chk_inv*:*ok*,*comfirm* 满足 Q 的需求.

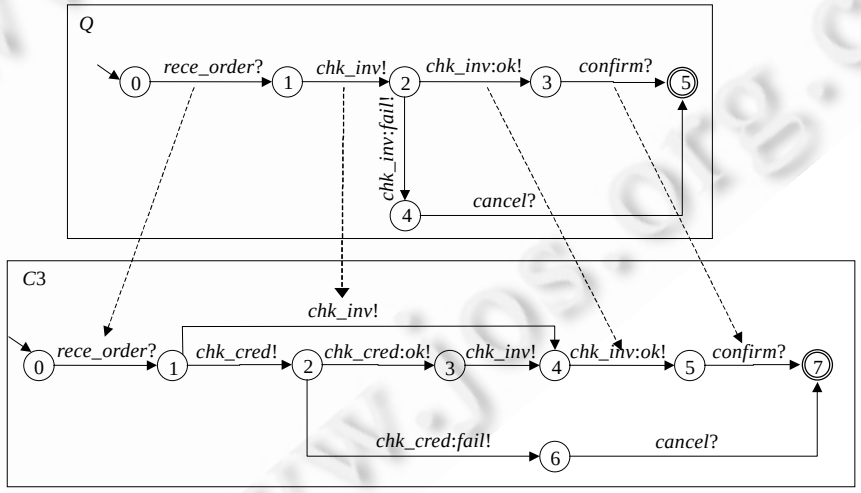

Fig.4 Compatible behavior matching

图 4 相容行为匹配

目前,基于行为的构件获取方法大都是是基于相容的行为匹配.为了支持更加松弛的行为匹配,本文提出了弱相容行为匹配、包含行为匹配和弱包含行为匹配的概念.

定义 7(出现). 设 Q 为一个需求构件,C 为构件库中的一个构件, $p'=a'_1,a'_2,...,a'_n \in L(C)$ 为 C 的一个字,如果在 Q 中存在一个动作序列 $p=a_1,a_2,\ldots,a_n \in L(Q)(m \leqslant n)$,$p$ 的一个子序列 $sp_i=a_{i1},a_{i2},\ldots,a_{im}(m \leqslant n)$满足条件 $a_{ij} \sim a'_j$ $(0 \leqslant j \leqslant m)$,则称 p'在 Q 中出现,记为 $p' \propto Q$.

定义 8(弱相容行为匹配). 设 Q 为一个需求构件,C 为构件库中的一个构件.如果存在字 $p'=a'_1,a'_2,...,a'_n \in L(C)$,满足条件 $p' \propto Q$,则称 C 为 Q 的一个弱相容的行为规约,记为 $C \xrightarrow{WComp\text{-}Beha} Q$.

根据弱相容行为匹配的定义,如果 $C \xrightarrow{WComp\text{-}Beha} Q$, 则说明 C 中至少存在 1 个字在行为上能够满足 Q 的部分需求.图 5 给出了一个弱相容行为匹配的实例.在本例中,构件 C_4 中的字:*rece_order*,*comfirm* 是 Q 的一个子动作序列.

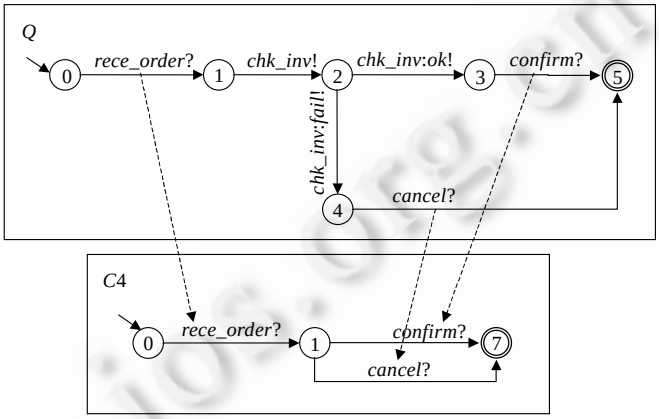

Fig.5 Weak compatible behavior matching

图 5 弱相容行为匹配

定义 9(包含子图). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,如果 P_Q 的一个子图 $SP_Q=(S_{S_Q},\Sigma_{S_Q},T_{S_Q},s_{S_{Q0}},S_{S_{QF}})$ 满足以下条件,则称 SP_Q 为 P_Q 的一个包含子图:

(1) $S_{S_Q} \subseteq S_Q, \Sigma_{S_Q} \subseteq \Sigma_Q, T_{S_Q} \subseteq T_Q, s_{S_{Q0}} \subseteq S_{S_Q}, S_{S_{QF}} \subseteq S_{S_Q}$.

(2) $NS(s_{S_{Q0}}) \subseteq S_{S_Q}, NT(s_{S_{Q0}}) \subseteq T_{S_Q}$.

(3) 对于任意 $s_{S_Q} \in S_{S_{QF}}$,有 $FS(s_{S_Q}) \subseteq S_{S_Q}, FT(s_{S_Q}) \subseteq T_{S_Q}$.

(4) 对于任意 $s_{S_Q} \in S_{S_Q} / \{s_{S_{Q0}}\} \cup S_{S_{QF}}$,有 $FS(s_{S_Q}) \subseteq S_{S_Q}, NS(s_{S_Q}) \subseteq S_{S_Q}, FT(s_{S_Q}) \subseteq T_{S_Q}, NT(s_{S_Q}) \subseteq T_{S_Q}$.

SP_Q 的初始状态是指没有前驱状态的状态,SP_Q 的终止状态是指没有后继状态的状态(初始状态除外,因为一个初始状态也可能是一个终止状态).

例如,图 6(a)为一个用户需求 Q,图 6(b)~图 6(e)给出了 P_Q 的 4 个子图,根据包含子图的定义,SP_{Q1} 和 SP_{Q4} 为 P_Q 的包含子图,SP_{Q2} 和 SP_{Q3} 不是 P_Q 的包含子图.在 SP_{Q2} 中,0 为初始状态,2 为终止状态.由于变迁 $0 \xrightarrow{d} 2 \notin T_{SQ2}$, 所以,$SP_{Q2}$ 不是 P_Q 的包含子图.在 SP_{Q3} 中,1 为初始状态,3 为终止状态,2 为内部状态.由于变迁 $0 \xrightarrow{d} 2 \notin T_{SQ3}$, 所以,$SP_{Q3}$ 不是 P_Q 的包含子图.

定义 10(包含行为匹配). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,C 为构件库中的一个构件,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.设 $SP_Q=(S_{S_Q},\Sigma_{S_Q},T_{S_Q},s_{S_{Q0}},S_{S_{QF}})$ 为 P_Q 的一个包含子图,如果存在一个从 S_{S_Q} 到 S_C 的二元关系 $f \subseteq S_{S_Q} \times S_C$ 满足如下条件,则称 C 为 Q 的一个包含行为规约,记为 $C \xrightarrow{Cont\text{-}Beha} Q$:

(1) 对于任意 $s_Q \in S_{S_Q}$,存在 $s_C \in S_C$,满足条件$(s_Q,s_C) \in f$.

(2) 对于 SQ 中的任意一个迁移 $s_Q \xrightarrow{a} s'_Q \in T_{S_Q}$,在 C 中存在一个迁移 $s_C \xrightarrow{a'} s'_C \in T_C$,满足条件:

$$(s_Q,s_C) \in f \wedge (s'_Q,s'_C) \in f \wedge (a \sim a').$$

(3) $(s_{S_{Q0}}, s_{C0}) \in f$.

(4) 对于任意 $s_{S_Q} \in S_{S_{QF}}$, 存在 $s_C \in S_{CF}$,满足条件 $(s_{S_Q}, s_C) \in f$.

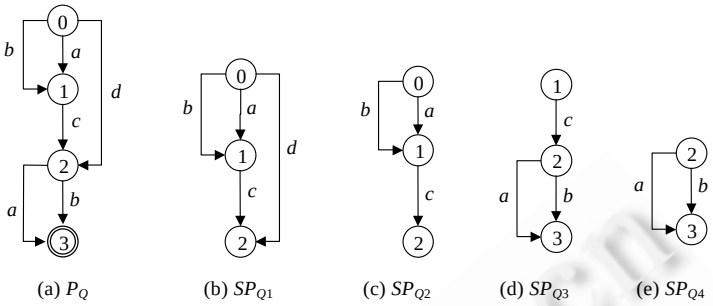

Fig.6 Instance of contain subgraph

图 6 包含子图实例

根据包含行为匹配的定义,如果 $C \xrightarrow{Cont\text{-}Beha} Q$, 则说明 C 在行为上可以实现 Q 的 1 个或多个包含子图的行为.图 7 给出了一个包含行为匹配的实例.在本例中,SP_Q 为 P_Q 的一个包含子图,其中,状态 1 为 SP_Q 的初始状态,状态 3 和状态 4 为 SP_Q 的终止状态.从 SP_Q 到 $C4$ 的映射关系为 f={(1,0),(2,1),(3,2),(4,2)}.

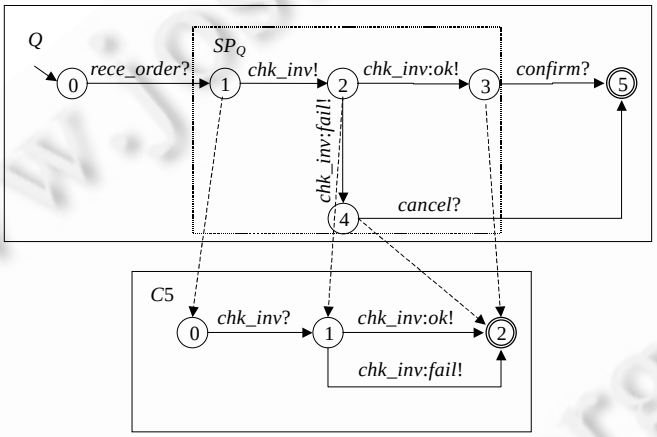

Fig.7 Contain behavior matching

图 7 包含行为匹配

定义 11(弱包含行为匹配). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,C 为构件库中的一个构件,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.设 $SP_Q=(S_{S_Q},\Sigma_{S_Q},T_{S_Q},s_{S_{Q0}},S_{S_{QF}})$ 为 P_Q 的一个包含子图,如果存在字 $p=a_1,a_2,\ldots,a_n \in L(SP_Q)$和 $p'=a'_1,a'_2,\ldots,a'_n \in L(C)$ 满足条件 $a_i \sim a'_i$ $(0 \leqslant i \leqslant n)$,则称 C 为 Q 的一个弱包含的行为规约,记为 $C \xrightarrow{WCont\text{-}Beha} Q$.

根据弱包含行为匹配的定义,如果 $C \xrightarrow{WCont\text{-}Beha} Q$, 则说明 C 至少存在 1 个运行动作序列在行为上满足 SP_Q 的需求.图 8 给出了一个弱包含行为匹配实例.在本例中,构件 $C6$ 中的字 *chk_inv*,*chk_inv*:*ok* 是 Q 的包含子图 SP_Q 的一个字.

在前述的 6 种构件行为匹配关系中,等价行为匹配的约束条件最强,弱相容行为匹配的约束条件最弱,扩展行为匹配的约束条件弱于等价行为匹配,但是强于包含行为匹配和相容行为匹配,包含行为匹配和相容行为匹配的约束条件强于弱包含行为匹配.图 9 给出了这 6 种构件行为匹配之间的蕴含关系,它们构成了一个多次的

构件行为匹配模型.如果构件 C 等价或者扩展行为匹配于用户需求 Q,则说明 C 能够完全满足 Q 的行为需求.否则,需要对 C 进行相应的扩展,或者对多个与 Q 满足(弱)包含、(弱)相容行为匹配的构件的组装来实现.

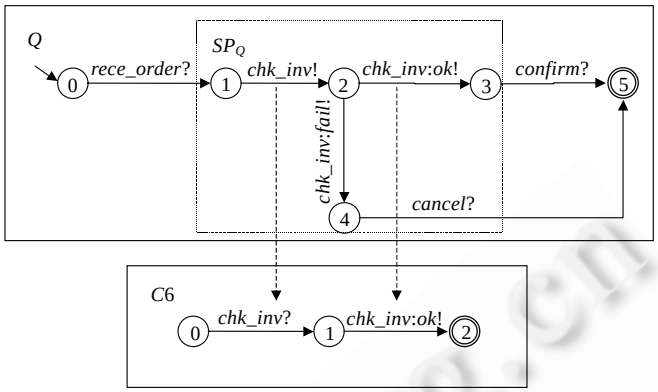

Fig.8 Weak contain behavior matching

图 8 弱包含行为匹配

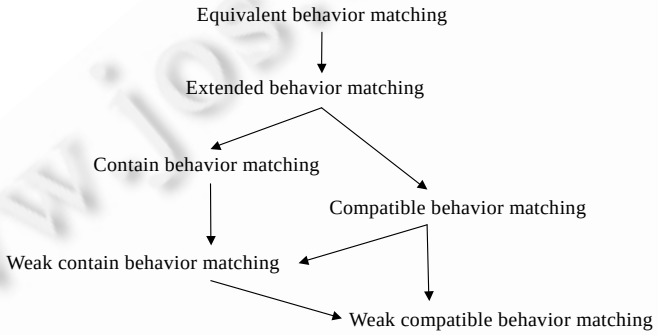

Fig.9 Multi-Level component behavior matching model

图 9 多层次构件行为匹配模型

4 行为匹配判定方法

本文使用从需求构件行为模型到构件行为模型的映射图来判定各种类型的行为匹配关系.

4.1 行为映射图

定义 12(行为映射图). 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型,C 为构件库中的一个构件,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.从 Q 到 C 的行为映射图可以定义为一个带有输入和输出的有限自动机,记为 $G_{QC}=(S,\Sigma,T,s_0,S_F)$.其中,

- $S=S_Q\times S_C$ 为 G_{QC} 的状态集,表示从 S_Q 到 S_C 的映射关系集.
- $\Sigma=\Sigma_Q\cup\Sigma_C$ 为 G_{QC} 所包含的动作集.
- $T=\{((s_Q,s_C),a/a',(s'_Q,s'_C))\mid(s_Q,a,s'_Q)\in T_Q,(s_C,a',s'_C)\in T_P,a\sim a'\}$ 为 G_{QC} 的状态迁移集,表示从 T_Q 到 T_C 的映射关系集.
- $s_0=(s_{Q0},s_{P0})$为 G_{QC} 的初始状态,表示从 s_{Q0} 到 s_{C0} 的映射关系.
- $S_F=S_{QF}\times S_{CF}\subseteq S$ 为 G_{QC} 的终止状态集,表示从 S_{QF} 到 S_{CF} 的映射关系集.

可以用 $s\xrightarrow{a/a'}s'$ 表示 G_{QC} 的一个迁移,其中,

$$s=(s_Q,s_C),\ s'=(s'_Q,s'_C)\in S,\ a\in\Sigma_Q,\ a'\in\Sigma_C,(s,a/a',s')\in T.$$

$s_Q\xrightarrow{a}s'_Q\in T_Q$ 为 $s\xrightarrow{a/a'}s'$ 在 Q 中所对应的迁移, $s_C\xrightarrow{a}s'_C\in T_C$ 为 $s\xrightarrow{a/a'}s'$ 在 C 中所对应的迁移.

定义 13(行为映射图的执行片段和运行). 行为映射图 G_{QC} 的一个执行片段可以定义为

$$\eta=s_0\xrightarrow{a_0/a'_0}s_1\xrightarrow{a_1/a'_1}s_2...s_{n-1}\xrightarrow{a_{n-1}/a'_{n-1}}s_n,$$

其中,$s_i=(s_{Qi},s_{Ci})(0\leqslant i\leqslant n)$, $s_i\xrightarrow{a_i/a'_i}s_{i+1}\in T$, s_0 为 G_{QC} 的初始状态.

如果 $s_n\in S_F$ 为 G_{QC} 的一个终止状态,则称 η 为 G_{QC} 的一个运行.

定义 14(输入和输出动作序列). 设 $\eta=s_0\xrightarrow{a_0/a'_0}s_1\xrightarrow{a_1/a'_1}s_2...s_{n-1}\xrightarrow{a_{n-1}/a'_{n-1}}s_n$ 为 G_{QC} 的一个执行片段,其中,$s_i=(s_{Qi},s_{Ci})\in S(0\leqslant i\leqslant n)$,则

(1) $\pi_Q(\eta)=s_{Q0}\xrightarrow{a_0}s_{Q1}\xrightarrow{a_1}s_{Q2}...s_{Q(n-1)}\xrightarrow{a_{n-1}}s_{Qn}$ 为 η 在 Q 上的投影,$\pi_Q(\eta)$ 所对应的动作序列为 η 所对应的输入动作序列,记为 $p_I(\eta)=a_0,a_1,\ldots,a_{n-1}$.

(2) $\pi_C(\eta)=s_{C0}\xrightarrow{a_0}s_{C1}\xrightarrow{a_1}s_{C2}...s_{C(n-1)}\xrightarrow{a_{n-1}}s_{Cn}$ 为 η 在 C 上的投影,$\pi_P(\eta)$ 所对应的动作序列为 η 所对应的输出动作序列,记为 $p_O(\eta)=a'_0,a'_1,...,a'_{n-1}$.

如果 η 为 G_{QC} 的一个运行,则称 $p_I(\eta)$ 为 η 所对应的输入运行动作序列,$p_O(\eta)$ 为 η 所对应的输出运行动作序列.可以用 $L_I(G_{QC})$ 和 $L_O(G_{QC})$ 分别表示 G_{QC} 的输入运行动作序列集和输出运行动作序列集.通过行为映射图 G_{QC},可以将 Q 中的每个运行动作序列映射到 C 中的相应运行动作序列上.

4.2 错误状态映射关系

在行为映射图 G_{QC} 中,并不是所有的状态映射关系都是正确的.下面,我们给出 G_{QC} 中存在的 9 种错误状态映射关系类型:

(1) 错误状态映射关系类型 E_1:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,如果在 Q 中存在一个迁移 $s_Q\xrightarrow{a}s'_Q\in T_Q$,但在 G_{QC} 中却不存在迁移 $(s_Q,s_C)\xrightarrow{a/a'}(s'_Q,s'_C)\in T$,则称 s 为错误状态映射关系类型 E_1.

(2) 错误状态映射关系类型 E_2:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,如果在 Q 中存在一个迁移 $s_Q\xrightarrow{a}s'_Q\in T_Q$,但在 G_{QC} 中却不存在迁移 $(s_Q,s_C)\xrightarrow{a/a'}(s'_Q,s'_C)\in T$;另外,在 Q 中存在另一个迁移 $s_Q\xrightarrow{a_1}s''_Q$ $(a_1\neq a)$,并且在 G_{QC} 中存在迁移 (s_Q,s_C) $(s_Q,s_C)\xrightarrow{a_1/a'_1}(s''_Q,s''_C)\in T$,则称 s 为错误状态映射关系类型 E_2.E_2 为 E_1 的子类型,即,如果 $s\in E_2$,则有 $s\in E_1$.

(3) 错误状态映射关系类型 E_3:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,如果 $s_Q\in S_{QF}\wedge s_C\notin S_{CF}$,则称 s 为错误状态映射关系类型 E_3.

(4) 错误状态映射关系类型 E_4:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个非初始状态,即 $s_Q\neq s_{Q0}\wedge s_C\neq s_{C0}$.如果 $FS(s)=\varnothing\vee FS(s)=\{s\}$,则称 s 为错误状态映射关系类型 E_4.

(5) 错误状态映射关系类型 E_5:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个非终止状态,即 $s_Q\notin S_{QF}\wedge s_C\notin S_{CF}$.如果 $NS(s)=\varnothing\vee NS(s)=\{s\}$,则称 s 为错误状态该映射关系类型 E_5.

(6) 错误状态映射关系类型 E_6:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个非初始状态,如果 $s_C\notin S_{CF}\wedge s\in E_1$,则称 s 为错误状态映射关系类型 E_6.E_6 是 E_1 的子类型,即,如果 $s\in E_6$,则 $s\in E_1$.

(7) 错误状态映射关系类型 E_7:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,如果 $s_C\notin S_{CF}\wedge s\in E_2$,则称 s 为错误状态映射关系类型 E_7.E_7 是 E_6 的子类型,即,如果 $s\in E_7$,则 $s\in E_6$.

(8) 错误状态映射关系类型 E_8:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,其中,$s_C\neq s_{C0}$.如果 $FS(s)=\varnothing\vee FS(s)=\{s\}$,则称 s 为错误状态映射关系 E_8.E_4 是 E_8 的子类型,如果 $s\in E_4$,则 $s\in E_8$.

(9) 错误状态映射关系类型 E_9:设 $s=(s_Q,s_C)\in S$ 为 G_{QC} 的一个状态,其中,$s_C\notin S_{CF}$.如果 $NS(s)=\varnothing\vee FS(s)=\{s\}$,则称 s 为错误状态映射关系 E_9.E_5 是 E_9 的子类型,如果 $s\in E_5$,则 $s\in E_9$.

4.3 扩展、相容、包含和弱包含行为匹配判定算法

基于行为映射图 G_{QC},我们可以判定需求构件 Q 与构件 C 之间的行为匹配关系.下面,我们给出一个可以判定扩展行为匹配、相容行为匹配、包含行为匹配和弱包含行为匹配的通用算法.

算法 1. 扩展、相容、包含和弱包含行为匹配判定算法.

JudgeSingleBehaMatch(**in** Q,**in** C,**in** *flag*,**inout** G_{QC},**out** R).

输入:

(1) Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型.C 为一个构件;

(2) $P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型;

(3) $G_{QC}=(S,\Sigma,T,s_0,S_F)$为从 Q 到 C 的行为映射图或行为映射子图;

(4) *flag*∈{*Exten-Beha*,*Comp-Beha*,*Cont-Beha*,*WCont-Beha*}为行为匹配类型标记:

(a) 如果 *flag=Exten-Beha*, 判定扩展行为匹配;

(b) 如果 *flag=Comp-Beha*, 判定相容行为匹配;

(c) 如果 *flag=Cont-Beha*, 判定包含行为匹配;

(d) 如果 *flag=WCont-Beha*, 判定弱包含行为匹配.

输出:

(1) 布尔变量 R;

(2) G_{QC} 的子图 SG_{QC}.

算法描述:

```
ES=∅;                                  //ES 中存储错误映射关系
do {
    for (each s∈S){
        if ((flag==Exten-Beha && s∈(E1∪E3∪E4∪E5))||
           (flag==Comp-Beha && s∈((E1/E2)∪E3∪E4∪E5))||
           (flag==Cont-Beha && s∈(E6∪E3∪E8∪E9))||
           (flag==WCont-Beha && s∈((E6/E7)∪E3∪E8∪E9)))
           Add(s,ES);                  //将 s 加入到 ES 中
    }
    if (ES!=∅){
            s=Get(ES);                 //从 ES 中取出一个映射关系 s
        if ((flag==Exten-Beha && s∈(E1∪E3))||
           (flag==Comp-Beha && s∈((E1/E2)∪E3))||
           (flag==Cont-Beha && s∈(E6∪E3))||
           (flag==WCont-Beha && s∈((E6/E7)∪E3))){
           DeleteFT(s,GQC);            //在 GQC 中删除 s 的所有前驱迁移
           DeleteNT(s,GQC);            //在 GQC 中删除 s 的所有后继迁移
        }
        if ((flag∈{Exten-Beha,Comp-Beha} && s∈E4)||
           (flag∈{Cont-Beha,WCont-Beha} && s∈E8))
           DeleteNT(s,GQC);            //在 GQC 中删除 s 的所有后继迁移
        if ((flag∈{Exten-Beha,Exten-Beha} && s∈E5)||
           (flag∈{Cont-Beha,WCont-Beha} && s∈E9))
           DeleteFT(s,GQC);            //在 GQC 中删除 s 的所有前驱迁移
```

```
            Delete(s,ES);                    //将 s 从 ES 中删除
            Delete(s,G_QC);                  //将 s 从 G_QC 中删除
        }
    } while (ES==∅)
    R=false;
        for (each s=(s_Q,s_C)∈S){
            if (flag∈{Exten-Beha,Comp-Beha} && (s_Q==s_Q0 && s_C==s_C0)){
          R=true;
              continue;
            }
            if (flag∈{Cont-Beha,WCont-Beha} && s_C==s_C0){
          R=true;
              continue;
            }
    }
    Output(G_QC);
    return R;
```

算法 1 的最坏情况时间复杂度为 $O(|S_Q|^2|S_C|^2)$,最好情况时间复杂度为 $O(|S_Q||S_C|)$.其中,$|S_Q|$表示 S_Q 中所包含的状态的数量,$|S_C|$表示 S_C 中所包含的状态的数量.

根据算法 1,我们可以判断 4 种构件行为匹配关系:扩展行为匹配(*flag=Exten-Beha*)、相容行为匹配(*flag=Comp-Beha*)、包含行为匹配(*flag=Cont-Beha*)和弱包含行为匹配(*flag=WCont-Beha*).其判定方法如下:

(1) 若要证明 $C \xrightarrow{Exten\text{-}Beha} Q$, 则需将 G_{QC} 中错误类型为 $E_1,E_3\sim E_5$ 的状态及其所关联的迁移从 G_{QC} 中删除,然后判断 G_{QC} 的初始状态是否存在:如果存在,则 $C \xrightarrow{Exten\text{-}Beha} Q$; 否则,$C$ 非扩展行为匹配于 Q.

(2) 若要证明 $C \xrightarrow{Comp\text{-}Beha} Q$, 则需将 G_{QC} 中错误类型为 $E_1/E_2,E_3\sim E_5$ 的状态及其所关联的迁移从 G_{QC} 中删除,然后判断 G_{QC} 的初始状态是否存在:如果存在,则 $C \xrightarrow{Comp\text{-}Beha} Q$; 否则,$C$ 非相容行为匹配于 Q.

(3) 若要证明 $C \xrightarrow{Cont\text{-}Beha} Q$, 则需将 G_{QC} 中错误类型为 E_6,E_3,E_8 和 E_9 的状态及其所关联的迁移从 G_{QC} 中删除,然后判断 G_{QC} 中是否存在形如(s_Q,s_{C0})的状态(其中,s_{C0} 为 C 的初始状态):如果存在,则 $C \xrightarrow{Cont\text{-}Beha} Q_P$; 否则,$C$ 非包含行为匹配于 Q.

(4) 若要证明 $C \xrightarrow{WCont\text{-}Beha} Q$, 则需将 G_{QC} 中错误类型为 $E_6/E_7,E_3,E_8$ 和 E_9 的状态及其所关联的迁移从 G_{QC} 中删除,然后判断 G_{QC} 中是否存在形如(s_Q,s_{C0})的状态(其中,s_{C0} 为 P 的初始状态):如果存在,则 $C \xrightarrow{WCont\text{-}Beha} Q$; 否则,$C$ 非弱包含行为匹配于 Q.

4.4 等价行为匹配判定算法

在 C 扩展行为匹配于 Q 的条件下,我们可以进一步判断 C 是否等价匹配于 Q.

下面,我们给出判断 $C \xrightarrow{Equi\text{-}Beha} Q$ 的算法.

算法 2. 等价行为匹配判定算法.

输入:用户需求 Q, $P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q_0},S_{QF})$ 为 Q 的行为模型;构件 C,$P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型.根据算法 1,在 *flag=Exten-Beha* 和 *R=true* 的情况下,获得从 Q 到 C 的行为映射子图 SG_{QC}.

输出:布尔变量 *R*.

算法描述:

步骤 1. 在 SG_{QC} 中检查 S_Q 与 S_C 之间是否为一一映射:如果为一一映射,则进入步骤 2;否则,*R=false*,算法结束.

步骤 2. 计算 SG_{QC} 在 C 上的投影 $\pi_C(SG_{QC})$.

步骤 3. 判断 $\pi_C(SG_{QC})$ 与 C 是否同构:如果同构,则 $R=true$;否则,$R=false$.

根据算法 2,如果 $R=true$,则 $C \xrightarrow{Equi\text{-}Beha} Q$; 否则,$C$ 非等价行为匹配于 Q.

在上述算法中,检查 S_Q 与 S_C 之间是否为一一映射的时间复杂度为 $O(|S_Q||S_{SG}|)$,计算 SG_{QC} 在 C 上的投影 $\pi_C(SG_{QC})$的时间复杂度为 $O(|S_{SG}|+|T_{SG}|)$,判断 $\pi_C(SG_{QC})$与 C 是否同构的时间复杂度为 $O(2^{S_{SG}})$.因此,算法 2 的时间复杂度为 $O(2^{S_{SG}})$.

4.5 弱相容行为匹配判定算法

本节讨论弱相容行为匹配的判定方法.根据弱相容行为匹配的定义,如果 C 中存在一个字出现在 Q 中,则 $C \xrightarrow{WComp\text{-}Beha} Q$. 由于 C 的行为模型中可能存在循环路径,因此 $L(C)$可能是无穷集合.此时,如果直接根据定义来判断 $C \xrightarrow{WComp\text{-}Beha} Q$ 是非常困难的.下面,我们给出简单运行和简单字的概念.

定义 15(简单运行和简单字). 设 $\eta = s_{C0} \xrightarrow{a_0} s_{C1} \xrightarrow{a_1} s_{C2}...s_{C(n-1)} \xrightarrow{a_{n-1}} s_{Cn} \in L(C)$ 为构件 C 的一个运行,如果 $s_{Ci} \neq s_{Cj}(i \neq j, 0 \leqslant i,j \leqslant n)$,则称 η 为 C 的一个简单运行,$p(\eta)=a_0,a_1,\ldots,a_{n-1}$ 为 C 的一个简单运行动作序列或简单字.我们用 $SP(C)$和 $SL(C)$分别表示 C 的简单运行集和简单字集.因为 C 的状态是有限的,所以 $SP(C)$和 $SL(C)$为有限集合.

定理 1. 设 $p \in L(C)$为 C 的一个字,如果 $p \propto Q$,则在 C 中一定存在一个简单字 $sp \in SL(C)$出现在 Q 中,即 $sp \propto Q$.

证明:如果 p 为一个简单字,则结论自然成立;否则,设 $\eta \in L(Q)$为 Q 的一个运行,$p=p(\eta)$为 η所对应的动作序列.由于 p 不是简单字,所以在 η中一定会出现重复的状态.我们可以按照下面的方法将 η转变为一个简单运行:

(1) 依次检查 η中的每个状态 $s_{pi}(0 \leqslant i \leqslant n)$.

(2) 从 sp_i 出发,查找 η中下一个与 sp_i 相等的状态 $sp_j(i \leqslant j)$.

(3) 合并 η中 sp_i 到 sp_j 的环为一个状态 sp_i.

根据上述方法,可以得到一个简单运行 η',并且 $p'=p(\eta')$是 p 的一个子序列.由于 $p \propto Q$,所以 $p' \propto Q$. □

根据定理 1,对于 C 的任何一个字,都存在一个简单字.因此,如要判断 $C \xrightarrow{WComp\text{-}Beha} Q$, 我们只需判断 $SP(C)$中是否存在简单字出现在 Q 上.如果出现,则 C 弱相容匹配于 Q;否则,C 非弱相容匹配于 Q.下面,我们通过一个定理给出判断一个字出现在 Q 中的方法.

定理 2. 设 Q 为一个需求构件,$P_Q=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$为 Q 的行为模型;C 为一个构件,$P=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$为 C 的行为模型;$G_{QC}=(S,\Sigma,T,s_0,S_F)$为从 Q 到 C 的映射图.设 $\eta \in P(C)$为 C 的一个运行,$p(\eta)$为 η所对应的字.如果在 G_{QC} 中存在一个有序迁移集,

$$\rho = \{(s_{Q0},s_{C0}) \xrightarrow{a_0/a'_0} (s_{Q1},s_{C1}),(s_{Q2},s_{C1}) \xrightarrow{a_2/a'_1} (s_{Q3},s_{C2}),\ldots,(s_{Q2(n-1)},s_{C(n-1)}) \xrightarrow{a_{2(n-1)}/a'_{n-1}} (s_{Q(2n-1)},s_{Cn})\} \in T,$$

其中,$(s_{Q2(i-1)},s_{C(i-1)}) \xrightarrow{a_{2(i-1)}/a'_{i-1}} (s_{Q(2i-1)},s_{Ci})$ $(0 \leqslant i \leqslant n)$为 G_{QC} 的一个迁移,满足条件:

$$\eta = s_{C0} \xrightarrow{a_0} s_{C1} \xrightarrow{a_1} s_{C2}...s_{C(n-1)} \xrightarrow{a_{n-1}} s_{Cn},$$

并且在 Q 中,$s_{Q2(i-1)}$到 s_{Q2i} 是可达的,则称 $p(\eta)$在 Q 上出现.其中,ρ称为 $p(\eta)$在 G_{QC} 中所对应的有序迁移集.

限于篇幅,证明略.

设 $\eta = s_{C0} \xrightarrow{a_0} s_{C1} \xrightarrow{a_1} s_{C2}...s_{C(n-1)} \xrightarrow{a_{n-1}} s_{Cn} \in L(C)$ 为 C 的一个运行,根据定理 2,如果在 G_{QC} 中存在 η的对应有序迁移集 ρ,则 $p(\eta) \propto Q$.下面给出判断 $p(\eta) \propto Q$ 的算法.

设 $\rho = \{(s_{Q0},s_{C0}) \xrightarrow{a_0/a'_0} (s_{Q1},s_{C1}),(s_{Q2},s_{C1}) \xrightarrow{a_2/a'_1} (s_{Q3},s_{C2}),\ldots,(s_{Q2(k-1)},s_{C(k-1)}) \xrightarrow{a_{2(k-1)}/a'_{k-1}} (s_{Q(2k-1)},s_{Ck})\}$为 G_{QC} 的一个有序迁移集,如果在 G_{QC} 中存在一迁移 $(s_{Q2k},s_{Ck}) \xrightarrow{a_{2k}/a'_k} (s_{Q(2k+1)},s_{C(k+1)})$,并且在 Q 中从 $s_{Q(2k-1)}$到 s_{Q2k} 是可达的,则称 $(s_{Q2k},s_{Ck}) \xrightarrow{a_{2k}/a'_k} (s_{Q(2k+1)},s_{C(k+1)})$为 η相对 ρ的可激活迁移集,令 $\rho_k(\eta)$为 η相对 ρ的可激活迁移集.

为了判断 $p(\eta) \propto Q$,我们首先令 $\rho=\varnothing$,$k=0$,然后采用先广度再深度的优先搜索方法依次检查有序迁移集 ρ的可激活迁移集 $\rho_k(\eta)(k=0,\ldots,n)$,并从 $\rho_k(\eta)$中选择一个未被访问过的迁移 $(s_{Q2k},s_{Ck}) \xrightarrow{a_{2k}/a'_k} (s_{Q(2k+1)},s_{C(k+1)})$,将其标记为访问过,并令 $\rho=\rho+\{(s_{Q2k},s_{Ck}) \xrightarrow{a_{2k}/a'_k} (s_{Q(2k+1)},s_{C(k+1)})\}$.如果 $k=n$,则 $p(\eta) \propto Q$,算法停止;否则,令 $k=k+1$.如果

$\rho_k(\eta)=\varnothing$,或者$\rho_k(\eta)$中的所有可激活迁移都已经被访问过,则令$\rho=\rho-\{(s_{Q2(k-1)},s_{C(k-1)})\xrightarrow{a_{2(k-1)}/a'_{k-1}}(s_{Q(2k-1)},s_{Ck})\}$,并令 $k=k-1$.如果 $k<0$,搜索过程终止;否则,重复上述过程.

算法 3. 判断字 $p(\eta)\in L(C)$是否出现在 Q 上.

输入:用户需求构件行为模型 $Q_P=(S_Q,\Sigma_Q,T_Q,s_{Q0},S_{QF})$,构件 C 的行为模型 $P_C=(S_C,\Sigma_C,T_C,s_{C0},S_{CF})$,字 $p(\eta)\in L(C)$,从 Q 到 C 的行为映射图 $G_{QC}=(S,\Sigma,T,s_0,S_F)$.

输出:布尔变量 R 和 $p(\eta)$所对应的有序迁移集ρ.如果 $R=true$,$p(\eta)$出现在 Q 上;否则,$p(\eta)$不出现在 Q 上.

算法描述:

```
ρ=∅; k=0;
构造集合ρ0(η);
if (ρ0(η)==∅) return false;
else for (each t∈ρ0(η)){在ρ0(η)中将 t 标记为未被访过}
do {
    if (ρk(η)!=∅ && ρk(η)中存在未被访问过的迁移){
       从ρk(η)中取出一个未被访问过的迁移(sQ2k,sCk) --a2k/a'k--> (sQ(2k+1),sC(k+1));
       在ρk(η)中将(sQ2k,sCk) --a2k/a'k--> (sQ(2k+1),sC(k+1))标记为访问过;
       ρ=ρ+{(sQ2k,sCk) --a2k/a'k--> (sQ(2k+1),sC(k+1));
    if (k==n){
       Output(ρ);
       return true;
    } else {
          k=k+1;
       构造集合ρk(η);
          for (each t∈ρk(η)){在ρk(η)中将 t 标记为未被访过}
    }
  } else {
       ρ=ρ-{(sQ2(k-1),sC(k-1)) --a2(k-1)/a'k-1--> (sQ(2k-1),sCk)};
       k=k-1;
  }
} while (k<0)
return false;
```

根据算法 3 可以判断 $p(\eta)$是否出现在 Q 上.当 $p(\eta)\propto Q$ 时,可以输出 $p(\eta)$在 G_{QC} 中所对应的有序迁移集ρ.上述算法的最好时间复杂度为 $O(|p(\eta)|)$,最坏时间复杂度为 $O(|T|)$.

为了判断 $C\xrightarrow{WComp\text{-}Beha}Q$, 我们需要依次判断 $SL(C)$中的每个简单字是否出现在 Q 上.

因此,判断 $C\xrightarrow{WComp\text{-}Beha}Q_p$ 的最好时间复杂度为 $O(|p(\eta_{\min})|)$,最坏时间复杂度为 $O((|T||SP(C)|))$.其中,$\eta_{\min}$为 $SP(C)$中长度最短的运行.

4.6 行为匹配判定算法

综合第 4.3 节~第 4.5 节所提出的算法,我们给出了一个可以判定用户需求与构件之间行为匹配关系的通用算法,其执行过程如图 10 所示.根据该算法,可以根据匹配之间的蕴含关系,给出需求构件与被检索构件之间最高级别的匹配关系.

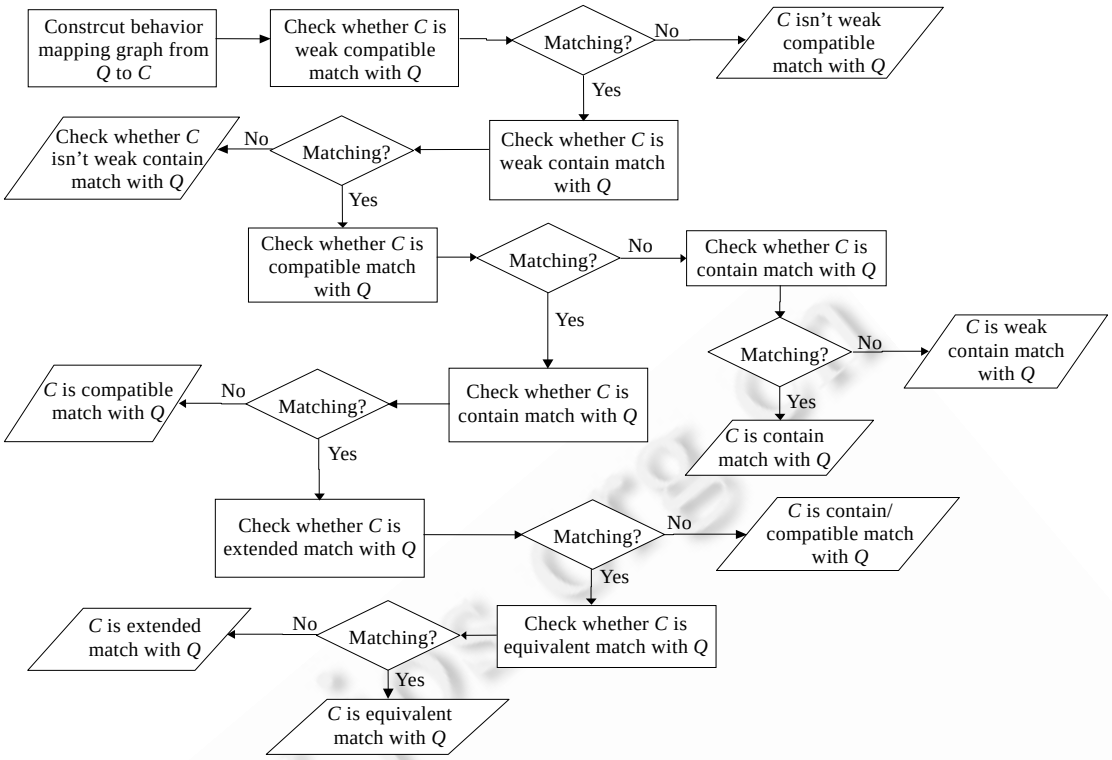

Fig.10 Process of judging behavior matching

图 10 行为匹配判定过程

5 基于行为的构件获取与适配

在基于行为的构件获取中,目前所提出的方法只考虑相容行为匹配[7,8,10,11],而没有考虑更加松弛的行为匹配,从而遗漏了许多可复用的构件,进而降低了构件的复用效率.如果采用本文提出的方法,对于给定的用户需求构件,如果构件库中不存在完全满足其行为需求的构件(即不存在等价或者扩展行为匹配的构件),但是存在 1 个或多个部分行为匹配的构件(即存在(弱)包含或(弱)相容行为匹配构件),则可以通过对其进行扩展或者通过多个构件的组装等适配技术来实现[15,16],从而提高了构件的复用效率.下面通过实例进行说明.

针对图 1 所示的需求构件 Q,假设构件库中存在如图 5、图 7 所示的两个构件 $C4$,$C5$,可以采用组装适配方法[15,17]对 $C4$,$C5$ 进行异步并行组装.组装后的复合构件扩展匹配于 Q,这样可以采用前期工作成果从中提取等价匹配于 Q 的行为[17].图 11 描述了组装后的复合构件,粗线部分给出等价匹配于 Q 的行为,它是由 $C4$ 和 $C5$ 协同完成的.

如果构件库中只存在单个包含匹配于 Q 的构件 $C5$,我们可以对 $C5$ 进行封装扩展.在其基础上包裹一个适配构件 A,A 的行为与 $C5$ 的行为综合作用的结果可以满足 Q 的需求.图 12 描述了适配的结果.由于 $C5$ 是 Q 的一个包含子图,因此在适配过程中其行为保持不变,从而有效提高了构件适配的效率.

如果构件库中只存在单个弱包含匹配于 Q 的构件 $C6$,我们可以采用文献[7]所提出的扩展适配方法,在 $C6$ 的基础上增加迁移 $1\xrightarrow{chk\_inv:\ fail}2$.适配后的构件包含匹配于 Q,从而可以采用上述方法进行封装扩展适配.

针对本文所提出的行为匹配模型,我们在 CERP 框架[18]的基础上设计了一个构件库管理原型系统,实现了上述的匹配思想.为了验证该方法的有效性,我们从 CERP 系统中选择了 100 个业务构件进行行为描述,让开发人员按照该方法进行业务构件获取.在构件检索中,如果用户指定具体某种匹配类型,则系统会查询出所有满足该匹配类型的业务构件;如果用户不指定具体的匹配类型,则系统会自动地检索出满足上述 6 种匹配关系的所

有构件,并给出相应的匹配类型和匹配结果.与目前只考虑相容行为匹配方法相比,本文所提出的方法具有较高的查全率和查准率.由于给出了相应的匹配类型和匹配结果,因此能够有效地指导后继构件的适配与组装.

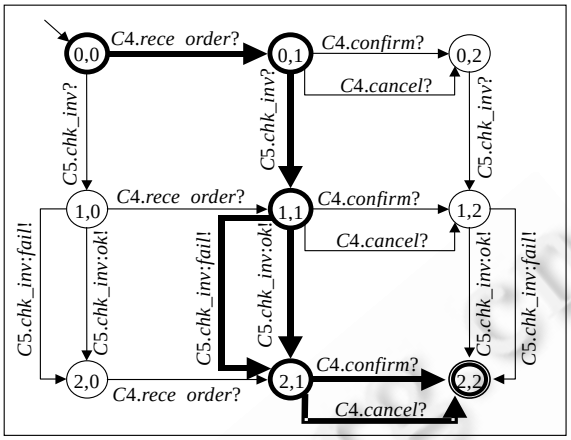

Fig.11 Composition component of $C4\times C5$

图 11 $C4\times C5$ 后的复合构件

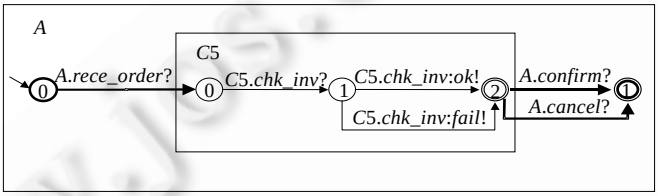

Fig.12 Adaptation of contain behavior matching component

图 12 包含行为匹配构件适配

6 总 结

本文针对基于接口自动机描述的构件行为模型,借鉴图匹配思想,提出了一种包含 6 种行为匹配关系的多层次行为匹配模型,并给出了一种可以判定各种行为匹配关系的通用判定方法以及相应的适配方法.与现有方法相比,该方法具有较高的查全率,且能有效地指导后继构件的适配与组装,从而提高了构件的复用效率.当然,由于行为匹配在效率方面存在一定问题,我们可以采用关键词或刻面的方法,与本文方法相结合,来提高查询效率.由于 Web 服务可以看作是一类特殊的构件,基于 BPEL 描述的组合服务可以很容易地转换为有限自动机[19].因此,本文所提出的方法也适用于基于行为的 Web 服务的获取.

References:

[1] Yang FQ, Lü J, Mei H. Technical framework for Internetware: An architecture centric approach. Science in China Sseries F—Information Sciences, 2008,51(6):610−622. [doi: 10.3724/SP.J.1001.2008.01201]

[2] Frakes WB. A case study of a reusable component collection in the information retrieval domain. Journal of Systems and Software, 2004,72(2):265−270. [doi: 10.1016/S0164-1212(03)00089-X]

[3] Lung CH, Urban JE. An approach to the classification of domain models in support of analogical reuse. ACM SIGSOFT Software Engineering Notes, 1995,20(8):169−178. [doi: 10.1145/211782.211842]

[4] Wang YF, Xue YJ, Zhang Y, Zhu SY, Qian LQ. A matching model for software component classified in faceted scheme. Journal of Software, 2003,14(3):401−408 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/401.htm

[5] Zaremski AM, Wing JM. Specification matching of software components. ACM Trans. on Software Engineering and Methodology, 1997,6(4):333−369. [doi: 10.1145/261640.261641]

[6] Alfaro L, Henzinger TA. Interface automata. In: Wermelinger M, Gall H, eds. Proc. of the 9th Annual ACM Symp. on Foundations of Software Engineering (FSE 2001). New York: ACM Press, 2001. 109−120.

[7] Redondo, PDR, Arias JJP, Vilas AF, Martinez BB. Approximate retrieval of incomplete and formal specifications applied to vertical reuse. In: Proc. of the Int'l Conf. on Software Maintenance (ICSM 2002). Los Alamitos: IEEE Computer Society, 2002. 618−627. [doi: 10.1109/ICSM.2002.1167826]

[8] Wombacher A, Fankhauser P, Mahleko B, Neuhold E. Matchmaking for business processes based on choreographies. In: Proc. of the 2004 IEEE Int'l Conf. on e-Technology, e-Commerce and e-Service (EEE 2004). Los Alamitos: IEEE Computer Society, 2004. 359−368.

[9] Grigori D, Corrales JC, Bouzeghoub M. Behavioral matchmaking for service retrieval. In: Proc. of the 2006 IEEE Int'l Conf. on Web Services (ICWS 2006). Los Alamitos: IEEE Computer Society, 2006. 145−152. [doi: 10.1109/ICWS.2006.37]

[10] Mahleko B, Wombacher A, Fankhauser P. A grammar-based index for matching business processes. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS 2005). Los Alamitos: IEEE Computer Society, 2006. 21−30. [doi: 10.1109/ICWS.2005.6]

[11] Hu JQ. Research on some key technologies of Web Service discovery [Ph.D. Thesis]. Changsha: Dissertation for Doctoral Degree of School of National University of Defense Technology, 2005 (in Chinese).

[12] Hu HY, Lü J, Ma XX, Tao XP. Study on behavioral compatibility of components in software architecture using object-oriented paradigm. Journal of Software, 2006,17(6):1276−1286 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1276.htm [doi: 10.1360/jos171276]

[13] Hameurlain N. A formal framework for component protocols behavioural compatibility. In: Proc. of the 13th Asia Pacific Software Engineering Conf. Los Alamitos: IEEE Computer Society, 2006. 87−94. [doi: 10.1109/APSEC.2006.4]

[14] Plasil F, Visnovsky S. Behavior protocols for software components. IEEE Trans. on Software Engineering, 2002,28(11): 1056−1076. [doi: 10.1109/TSE.2002.1049404]

[15] Schmidt HW, Reussner RH. Automatic component adaptation by concurrent state machine retrofitting. Technical Report, 25/2000, Universität Karlsruhe, Department of Informatics, 2000.

[16] Reussner RH. Automatic component protocol adaptation with the CoConut/J tool suite. Future Generation Computer System, 2003, 19(5):627−639. [doi: 10.1016/S0167-739X(02)00173-5]

[17] Meng FC. Development method of model driven and component based enterprise software and application [Ph.D. Thesis]. Harbin: Dissertation for Doctoral Degree of Harbin Institute of Technology, 2008 (in Chinese).

[18] Hu LL. Application framework based on MVC design pattern [MS. Thesis]. Harbin: Dissertation for Master Degree of Harbin Institute of Technology, 2006 (in Chinese).

[19] Wombacher A, Fankhauser P, Neuhold E. Transforming BPEL into annotated deterministic finite state automata for service discovery. In: Proc. of the IEEE Int'l Conf. on Web Services (ICWS 2004). Los Alamitos: IEEE Computer Society, 2004. 316−323. [doi: 10.1109/ICWS.2004.1314753]

附中文参考文献:

[4] 王渊峰,薛云皎,张涌,朱三元,钱乐秋.刻面分类构件的匹配模型.软件学报,2003,14(3):401−408. http://www.jos.org.cn/1000-9825/14/401.htm

[11] 胡建强.Web 服务发现若干关键技术研究[博士学位论文].长沙:国防科学技术大学,2005.

[12] 胡海洋,吕建,马晓星,陶先平.面向对象范型体系结构中构件行为相容性研究.软件学报,2006,17(6):1276−1286. http://www.jos.org.cn/1000-9825/17/1276.htm [doi: 10.1360/jos171276]

[17] 孟凡超.模型驱动的构件化企业应用软件开发方法[博士学位论文].哈尔滨:哈尔滨工业大学,2008.

[18] 胡伶俐.基于 MVC 的新型 CERP 应用程序框架的研究与实现[硕士学位论文].哈尔滨:哈尔滨工业大学,2006.

初佃辉(1970－),男,山东潍坊人,博士生,副教授,CCF 会员,主要研究领域为企业资源计划,电子商务,供应链管理,物流管理.

战德臣(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为数据库与知识工程,软件重构与复用,虚拟企业集成与电子商务,企业资源计划系统.

孟凡超(1974－),男,博士,副教授,CCF 会员,主要研究领域为模型驱动的体系结构,软件重构与复用,企业资源计划.

徐晓飞(1962－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为企业智能计算,服务计算,管理与决策信息系统,数据库,企业资源计划与供应链管理技术,电子商务与商务智能.